DOCUMENTATION FOR THE FRAMES-HWIR TECHNOLOGY SOFTWARE SYSTEM, VOLUME 13: CHEMICAL PROPERTIES PROCESSOR

Project Officer and Technical Direction:

Mr. Gerard F. Laniak
U.S. Environmental Protection Agency
Office of Research and Development
National Environmental Research Laboratory
Athens, Georgia 30605

Prepared by:

Pacific Northwest National Laboratory
Battelle Boulevard, P.O. Box 999
Richland, Washington 99352
Under EPA Reference Number DW89937333-01-0

U.S. Environmental Protection Agency
Office of Research and Development
Athens, Georgia 30605

October 1999

DISCLAIMER

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC06-76RLO 1830

Acknowledgments

A number of individuals have been involved with this effort. Mr. Gerard F. Laniak of the U.S. Environmental Protection Agency (EPA), Office of Research and Development, National Environmental Research Laboratory, Athens, Georgia, provided the overall technical direction and review throughout this work. This report was prepared by the Pacific Northwest National Laboratory[1] (PNNL) staff of Karl Castleton, Regina Lundgren, Gene Whelan, Gariann Gelston, Bonnie Hoopes, John McDonald, Mitch Pelton, and Randal Taira. Additional PNNL staff supporting this effort include Wayne Cosby, Nancy Foote, Kristin Manke, Jill Pospical, Debbie Schulz, and Barbara Wilson. Useful inputs were provided by many U.S. EPA individuals working on the Hazardous Waste Identification Rule, including Messrs. Barnes Johnson, Stephen Kroner, and David Cozzie, and Drs. David Brown, Robert Ambrose, Zubair Saleem, Donna Schwede, and Sharon LeDuc, among many others.

[1]Operated by Battelle for the U.S. Department of Energy under Contract DE-AC06-76RLO 1830.

Summary

The U.S. Environmental Protection Agency (EPA) is developing a comprehensive environmental exposure and risk analysis software system for agency-wide application. The software system will be applied to the technical assessment of exposures and risks relevant to the Hazardous Waste Identification Rule (HWIR). The software system adapted to automate this assessment is the Framework for Risk Analysis in Multimedia Environmental Systems (FRAMES), developed by the Pacific Northwest National Laboratory. The process used to develop the FRAMES-HWIR Technology Software System includes steps for requirements analysis, design, specification, and development with testing and quality assurance comprising a critical portion of each step. This report documents that process for one of the components of the system: the Chemical Properties Processor (CPP). This processor will

1) Compute specified chemical properties using mathematical relationships, as presented in Section 2.2, to documents provided in EPA Document "HWIR Chemical Database."

2) Compute certain chemical properties using a statistical random sampling process.

3) Read an Organic Chemical Property data table that is stored as a flat-ASCII file and populated by EPA.

4) Read a Metal/Inorganic Chemical Property data table that is stored as a flat-ASCII file and populated by EPA.

5) Read the Transformation Products data tables that are stored as flat-ASCII files and populated by EPA. The Transformation Products tables consist of seven tables: 1) Catalyzation, 2) Aerobic Biodegradation, 3) Activated Biodegradation, 4) Anaerobic Biodegradation, 5) Anaerobic Reduction Biodegradation, 6) SO_4 Reduction Biodegradation, and 7) Methanogenic Biodegradation data tables.

6) Read a Human Health Benchmarks data table that is stored as a flat-ASCII file and populated by EPA.

7) Read an Ecological Benchmarks data table that is stored as a flat-ASCII file and populated by EPA.

8) Read an Ecological Bioaccumulation Factors data table that is stored as a flat-ASCII file and populated by EPA.

9) Read an Aquatic Bioaccumulation Factors data table that is stored as a flat-ASCII file and populated by EPA.

10) Read a Chemical Ecological Flag data table that is stored as a flat-ASCII file and populated by EPA.

11) Read a Waste Concentration data table that is stored as a flat-ASCII file and populated by EPA.

12) Be testable as a stand-alone processor.

The CPP facilitates the computation of chemical properties used by other components within the FRAMES-HWIR Technology Software System, in particular, the Site Definition Processor (SDP) and modules within the Multimedia Multipathway Simulation Processor (MMSP). The CPP creates the chemical site definition files for the chemical specified by the SDP. Modules within the MMSP use the CPP to gather chemical data necessary to simulate specific environmental media interactions. The System User Interface (SUI) also uses the CPP to identify which chemicals have sufficient data to be simulated in the HWIR assessment and to provide the user with a list of chemicals to use in the simulation. This report includes information on requirements of the CPP and design elements necessary to meet those requirements. It also discusses testing plans, testing results, and the quality assurance program for the CPP.

Acronyms and Abbreviations

ABF	Aquatic Bioaccumulation Factors
ActBio	Activated Biodegradation
AerBio	Aerobic Biodegradation
AnaBio	Anaerobic Biodegration
AnaRedBio	Anaerobic Reduction Biodegradation
ASCII	American Standard Code for Information Interchange
CASID	Chemical Abstract System Identification
CAT	Catalyzation
CPP	Chemical Properties Processor
CP.SSF	Chemical Properties Site Simulation File
EB	Ecological Benchmarks
EBF	Ecological Bioaccumulation Factors
EOF	end of file
EPA	U.S. Environmental Protection Agency
FRAMES	Framework for Risk Analysis in Multimedia Environmental Systems
GRF	Global Results Files
HHB	Human Health Benchmarks
HWIR	Hazardous Waste Identification Rule
MET	meteorological
MethBio	Methanogenic Biodegradation
MICP	Metal/Inorganic Chemical Property
MMSP	Multimedia Multipathway Simulation Processor
OCP	Organic Chemical Property
PNNL	Pacific Northwest National Laboratory
SDP	Site Definition Processor
SO4Bio	SO_4 Reduction Biodegradation
SUI	System User Interface
TP	Transformation Products

Contents

Figures

Tables

1.0 Introduction

The U.S. Environmental Protection Agency (EPA) is developing a comprehensive environmental exposure and risk analysis software system for agency-wide application. The software system will be applied to the technical assessment of exposures and risks relevant to the Hazardous Waste Identification Rule (HWIR). The HWIR is designed to determine quantitative criteria for allowing a specific class of industrial waste streams to no longer require disposal as a hazardous waste (that is, allow such streams to "exit" Resource Conservation and Recovery Act [RCRA], Subtitle C) and allow RCRA disposal in Subtitle D facilities as industrial waste. Hazardous waste constituents with values less than these exit criteria levels would be reclassified as nonhazardous wastes under the Resource Conservation and Recovery Act.

The software system adapted to automate this assessment is the Framework for Risk Analysis in Multimedia Environmental Systems (FRAMES), developed by the Pacific Northwest National Laboratory (PNNL). The FRAMES-HWIR Technology Software System consists of a series of components within a system framework (Figure 1.1). The process used to develop the FRAMES-HWIR Technology Software System includes steps for requirements analysis, design, specification, and development, with testing and quality assurance comprising a critical portion of each step.

This report discusses one of the components of the system: the Chemical Properties Processor (CPP). This processor facilitates the computation of chemical properties used by other components within the FRAMES-HWIR Technology Software System, primarily the Site Definition Processor (SDP) and modules within the Multimedia Multipathway Simulation Processor (MMSP). The SDP uses the CPP to create the Site Definition Files, which will contain control information and a conceptual site model that describes the combination of multimedia modules that are to be executed for the simulation. The modules within the MMSP use the CPP to gather chemical data necessary to simulate specific environmental media interactions. The System User Interface (SUI) also uses the CPP to identify which chemicals have sufficient data to be simulated in the HWIR assessment and to provide the user with a list of chemicals to use in the simulation.

The CPP is being designed as a dynamic link library (DLL) to facilitate consistent communication between components. A DLL is a modular set of routines that comes with or can be added to a software system. Each DLL file has a ".dll" file name extension. DLL files are dynamically linked with the program that uses them during program execution instead of being compiled with the main program. The set of such files is somewhat comparable to the library routines provided with programming languages such as FORTRAN, C, and C++.

This report includes information on requirements of the CPP and design elements necessary to meet those requirements. It also discusses testing plans, testing results, and the quality assurance program for the CPP. Specifications for the CPP are described in *Documentation of the FRAMES-HWIR Technology Software System, Volume 8: Specifications*. References cited in the text are listed in Section 6.0. Appendix A provides additional details on the testing program for the CPP. Other components developed by PNNL are described in companion documents as listed in the reference list; the system itself is documented in a summary report entitled,*Overview of the FRAMES-HWIR Technology Software System.*

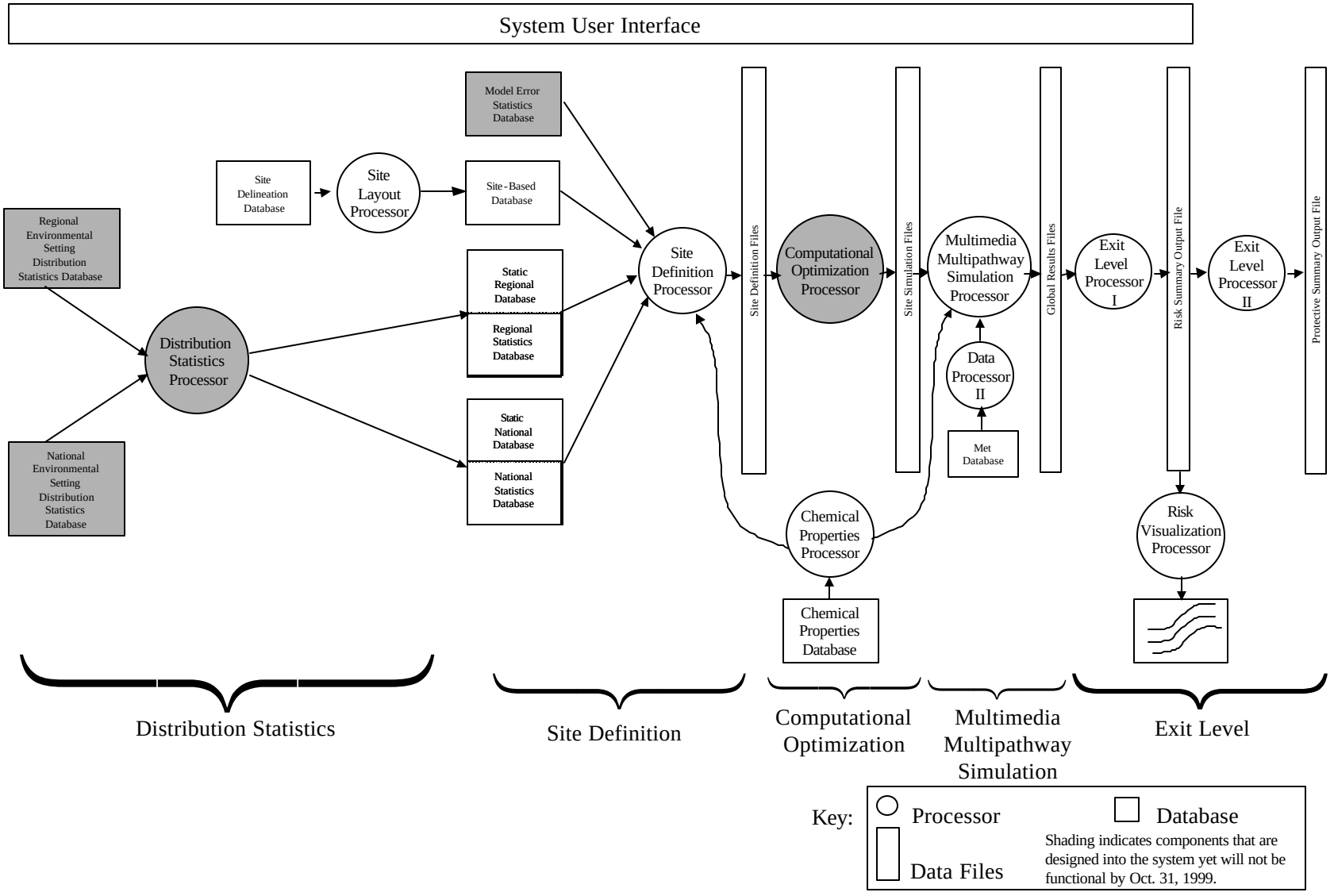

Figure 1.1 Overview of the FRAMES-HWIR Technology Software System

2.0 Requirements

Requirements are characteristics and behaviors that a piece of software must possess to function adequately for its intended purpose. As mentioned, the purpose of the CPP is to calculate chemical properties for various components of the FRAMES-HWIR Technology Software System. In summary, the CPP will

1) Compute specified chemical properties using mathematical relationships, as presented in Section 2.2 to documents provided in EPA Document "HWIR Chemical Database."

2) Generate certain chemical properties using a statistical random sampling process.

3) Read an Organic Chemical Property (OCP) data table that is stored as a flat-ASCII file and populated by EPA. The format of the OCP data table is defined in *Documentation of the FRAMES-HWIR Technology Software System, Volume 8: Specifications*.

4) Read a Metal/Inorganic Chemical Property (MICP) data table that is stored as a flat-ASCII file and populated by EPA. The format of the MICP data table is defined in *Documentation of the FRAMES-HWIR Technology Software System, Volume 8: Specifications*.

5) Read the Transformation Products (TP) data tables that are stored as flat-ASCII files and populated by EPA. The TP tables consist of seven tables: 1) Catalyzation (CAT), 2) Aerobic Biodegradation (AerBio), 3) Activated Biodegradation (ActBio), 4) Anaerobic Biodegration (AnaBio), 5) Anaerobic Reduction Biodegradation (AnaRedBio), 6) SO_4 Reduction Biodegradation (SO4Bio), and 7) Methanogenic Biodegradation (MethBio) data tables. The format of the TP data tables is defined in *Documentation of the FRAMES-HWIR Technology Software System, Volume 8: Specifications*.

6) Read a Human Health Benchmarks (HHB) data table that is stored as a flat-ASCII file and populated by EPA. The format of the HHB data table is defined in *Documentation of the FRAMES-HWIR Technology Software System, Volume 8: Specifications*.

7) Read an Ecological Benchmarks (EB) data table that is stored as a flat-ASCII file and populated by EPA. The format of the EB data table is defined in *Documentation of the FRAMES-HWIR Technology Software System, Volume 8: Specifications*.

8) Read an Ecological Bioaccumulation Factors (EBF) data table that is stored as a flat-ASCII file and populated by EPA. The format of the EBF data table is defined in *Documentation of the FRAMES-HWIR Technology Software System, Volume 8: Specifications*.

9) Read an Aquatic Bioaccumulation Factors (ABF) data table that is stored as a flat-ASCII file and populated by EPA. The format of the ABF data table is defined in *Documentation of the FRAMES-HWIR Technology Software System, Volume 8: Specifications*.

10) Read a Chemical Ecological Flag data table that is stored as a flat-ASCII file and populated by EPA. The format of this data table is defined in *Documentation of the FRAMES-HWIR Technology Software System, Volume 8: Specifications*.

11) Read a Waste Concentration data table that is stored as a flat-ASCII file and populated by EPA. The format of this data table is defined in *Documentation of the FRAMES-HWIR Technology Software System, Volume 8: Specifications*.

12) Be testable as a stand alone processor.

The following subsections describe in further detail the input, scientific, and output requirements of the CPP.

2.1 Input Requirements

As noted above, the CPP.DLL will be used by the SUI, SDP, and modules within the MMSP. Subroutines within these processors and modules will call subroutines within the CPP.DLL. To facilitate the subroutine calls, the CPP.DLL will read the chemical database, which is a specified set of formatted ASCII files. The term formatted ASCII refers to a text file that contains only 7-bit ASCII characters and uses only ASCII-standard control characters (that is, has no embedded codes specific to a particular text formatter markup language or output device and no meta-characters). Additional information on the formatted-ASCII file can be found in *Documentation for the FRAMES-HWIR Technology Software System, Volume 8: Specifications* (see Section 6.0, References).

The CPP.DLL recognizes only the following data tables:

1. OCP
2. MICP
3. Transformation Products (TP)
   - 3.a CAT
   - 3.b AerBio
   - 3.c ActBio
   - 3.d Ana Bio
   - 3.e AnaRedBio
   - 3.f SO4Bio
   - 3.g MethBio
4. HHB
5. EB
6. EBF
7. ABF
8. Chemical Ecological Flag
9. Waste Concentration

Each file format is defined by the column definitions of the data. The data files will be represented by a flat-ASCII, comma-separated values file that has the following general format:

1. After the four header lines, the fifth line will have the number of rows in the data table and the number of columns in the data table.
2. The next line will contain column names.
3. The next line will contain column units.
4. The next line will contain column data types.
5. The next line through the end of file (EOF) will contain the rows of information.

A small example would be as follows:

```
5,3
“CHName"        ,”VolumeA"      ,”VolumeB"
                ,”mL”           ,”mL”
“String(32)”    ,”Real”         ,”Real”
“Ammonia”       ,               ,
“Acrylic Acid”  ,4.96e+1        ,6.65e-2
“Acetamide”     ,4.19e+1        ,6.16e-2
“Acenapthene”   ,1.22e+2        ,6.75e-2
“Acetic Acid”   ,3.91e+1        ,6.16e-2
```

2.2 Scientific Requirements

See Section 2.3 for a discussion of output requirements that include scientific calculations. Development of the CPP.DLL assumes that only two languages will need to be supported: C++ and FORTRAN 77 (or later versions). The only limitation of the CPP.DLL is that it is a 32-bit DLL that must be run under Windows 95.

2.3 Output Requirements

The CPP is required to perform the following computations on request:

1) Compute a statistically sampled value associated with a chemical property from a distribution. The distribution types available for selection, as part of the statistical sampling procedure, are limited to those associated with the sampling algorithm. Unless available through the sampling algorithm, no other distribution type will be used by the CPP.

2) Compute values for pH- and temperature-dependent organic chemical characteristics, using algorithms defined by the EPA document “HWIR Chemical Database.”

3.0 Design Elements

Design elements are strategies for meeting requirements. The key design elements of the CPP.DLL are the initialization and input subroutines, which are detailed below for two interfaces: FORTRAN and C++. The C++ interfaces are also compatible with versions of C that support interfaces or prototypes. Both languages are needed because other modules within the FRAMES-HWIR Technology Software System are programmed in these languages. Note that only those subroutines and functions that are passed a number of arguments have associated tables of information. Section 3.1 describes subroutines for initializing the CPP.DLL, and Section 3.2 describes calculational subroutines and functions included in the CPP.DLL.

3.1 Initialization Subroutines

The following two subroutines initialize the CPP.DLL.

3.1.1 Subroutine ChemEnv

This subroutine initializes the environmental parameters for the CPP. DLL for functions that need to make assumptions about the acidity, alkalinity, and neutral conditions of the media. If the pH is less than 6, acidic conditions will be assumed. If the pH is greater than 8, base conditions will be assumed. Values between 6 and 8 inclusively will be assumed to be neutral conditions.

FORTRAN Interface

Subroutine ChemEnv(Temperature,pH,Media)

C++ Prototype

void ChemEnv(Temperature,pH,Media);

Arguments	**FORTRAN Type**	**C++ Type**
Temperature	Real*8	double
pH	Real*8	double
Media (choice of soil, sediment, or surface water)	Character*32	char *

3.1.2 Subroutine ChemCASID

This subroutine initializes the CPP to retrieve values for a selected chemical, based on the Chemical Abstract System Identification (CASID).

FORTRAN Interface

Subroutine ChemCASID(CASID)

C++ Prototype

void ChemCASID(CASID);

Arguments	FORTRAN Type	C++ Type
CASID (a valid CASID that contains dashes)	Character*32	char *

3.1.3 Subroutine ChemPath

This subroutine initializes the CPP to retrieve values from chemical data tables at a specified directory or path.

FORTRAN Interface

Subroutine ChemPath(Path)

C++ Prototype

void ChemPath(Path);

Arguments	FORTRAN Type	C++ Type
Path to CPP data tables	Character*32	char *

3.2 Input Subroutines and Functions

The following subroutines and functions allow the CPP.DLL to return information that is used as input to the SDP, SUI, or MMSP modules.

3.2.1 Function NumChem

This function returns the integer value for the number of valid chemicals for which the CPP has properties. If a chemical is missing part of its required data, it will not be included in this count.

FORTRAN Interface

Function NumChem() Integer

C++ Prototype

int NumChem();

3.2.2 Subroutine ChemInfo

This subroutine is used in conjunction with Function NumChem to create a list of the available chemicals (and their associated index, name, and CASID) in the database the CPP.DLL is reading. ChemType is set to an integer value that represents the type of the chemical (1=Organic and 2=Metal/Inorganic).

FORTRAN Interface

Subroutine ChemInfo(Index,Name,CASID,ChemType)

C++ Prototype

void ChemInfo(Index,Name,CASID,ChemType)

Arguments	FORTRAN Type	C++ Type
Index	Integer	int
Name	Character*32	char*
CASID	Character*32	char*
ChemType	Integer	int *

3.2.3 Subroutine SMILES

This subroutine returns the SMILES string for organic chemicals.

FORTRAN Interface
subroutine ChemPhysProp(SMILES)
C++ Prototype
void ChemPhysProp(SMILES)

Arguments	FORTRAN Type	C++ Type
SMILES	Character*80	char*

3.2.4 Function ChemADiff

This function computes the air diffusion coefficient for organic chemicals in cm^2/s.

FORTRAN Interface
function ChemADiff() Real*8
C++ Prototype
double ChemADiff()

3.2.5 Function ChemVol

This function computes the volume for organic chemicals in mL.

FORTRAN Interface
function ChemVol() Real*8
C++ Prototype
double ChemVol()

3.2.6 Function ChemDen

This function computes the density for organic chemicals in g/mL.

FORTRAN Interface
function ChemDen() Real*8
C++ Prototype
double ChemDen()

3.2.7 Function ChemWDiff

This function computes the water diffusion coefficient for organic chemicals, in cm^2/s.

FORTRAN Interface
function ChemWDiff() Real*8
C++ Prototype
double ChemWDiff()

3.2.8 Function ChemVP

This function computes the vapor pressure for organic chemicals in torr.

FORTRAN Interface
function ChemVP() Real*8
C++ Prototype
double ChemVP()

3.2.9 Function ChemSol

This function computes the solubility limit for organic chemicals in mg/L. It samples a solubility limit value from a distribution for metals and inorganics using a statistical random sampling process.

FORTRAN Interface
function ChemSol() Real*8
C++ Prototype
double ChemSol()

3.2.10 Function ChemHLC

This function computes the Henry's Law Constant for organic chemicals; in (atm m^3 / mol).

FORTRAN Interface
function ChemHLC() Real*8
C++ Prototype
double ChemHLC()

3.2.11 Function ChemKow

This function computes the Kow for organic chemicals (dimensionless).

FORTRAN Interface
function ChemKow() Real*8
C++ Prototype
double ChemKow()

3.2.12 Function ChemKoc

This function computes the Koc for organic chemicals in mL/g.

FORTRAN Interface
function ChemKoc() Real*8
C++ Prototype
double ChemKoc()

3.2.13 Function ChemHyd

This function computes the hydrolysis rate for organic chemicals in 1/day units.

FORTRAN Interface
function ChemHyd() Real*8
C++ Prototype
double ChemHyd()

3.2.14 Function ChemKd

This function computes the partition coefficient for metals/inorganic chemicals in L/kg.

FORTRAN Interface
function ChemKd() Real*8
C++ Prototype
double ChemKd()

3.2.15 Subroutine ChemCat

This subroutine returns the catalyzed hydrolysis rate constant (Rate in 1/day units) and number of reaction products (NumProd). The determination of acid, neutral, or base conditions is made from the value set for pH (see description of Subroutine ChemEnv, Section 3.1.1).

FORTRAN Interface
Subroutine ChemCat(Rate,NumProd)
C++ Prototype
void ChemCat(Rate,NumProd)

Arguments	FORTRAN Type	C++ Type
Rate (1/day)	Real*8	Double *
NumProd	Integer*2	int *

3.2.16 Subroutine ChemPCat

This subroutine returns data about a particular product that is associated with the catalyzed hydrolysis rate constant. The name (Name), product CASID (PCASID), and molar yield coefficient for reaction (YCoef) in moles/mole are returned. The determination of acid, neutral, or base conditions is made from the value set for pH (see description of Subroutine ChemEnv, Section 3.1.1).

FORTRAN Interface

Subroutine ChemPCat(Index,Name,PCASID,YCoef)

C++ Prototype

void ChemPCat(Index,Name,PCASID,YCoef)

Arguments	FORTRAN Type	C++ Type
Index	Integer*2	int
Name	Character*32`	char*
PCASID	Character*32	char*
YCoef(moles/mole)	Real*8	Double *

3.2.17 Subroutine ChemAerBio

This subroutine returns the aerobic biodegradation rate constant (Rate) in 1/day units and number of reaction products (NumProd).

FORTRAN Interface

Subroutine ChemAerBio(Rate,NumProd)

C++ Prototype

void ChemAerBio(Rate,NumProd)

Arguments	FORTRAN Type	C++ Type
Rate (1/day)	Real*8	Double *
NumProd	Integer*2	int *

3.2.18 Subroutine ChemPAerBio

This subroutine returns data about a particular product that is associated with the aerobic biodegradation rate constant. The name (Name), product CASID (PCASID), and molar yield coefficient for reaction (YCoef) in moles/mole are returned.

FORTRAN Interface

Subroutine ChemPAerBio(Index,Name,PCASID,YCoef)

C++ Prototype

void ChemPAerBio(Index,Name,PCASID,YCoef)

Arguments	FORTRAN Type	C++ Type
Index	Integer*2	int
Name	Character*32	char*
PCASID	Character*32	char*
YCoef(moles/mole)	Real*8	Double*

3.2.19 Subroutine ChemActBio

This subroutine returns the activated biodegradation rate constant (Rate) in 1/day units and number of reaction products (NumProd).

FORTRAN Interface

Subroutine ChemActBio(Rate,NumProd)

C++ Prototype

void ChemActBio(Rate,NumProd)

Argument	FORTRAN Type	C++ Type
Rate (1/day)	Real*8	Double *
NumProd	Integer*2	int *

3.2.20 Subroutine ChemPActBio

This subroutine returns data about a particular product that is associated with the activated biodegradation rate constant. The name (Name), product CASID (PCASID), and molar yield coefficient for reaction (YCoef) in moles/mole are returned.

FORTRAN Interface

Subroutine ChemPActBio(Index,Name,PCASID,YCoef)

C++ Prototype

void ChemPActBio(Index,Name,PCASID,YCoef)

Arguments	FORTRAN Type	C++ Type
Index	Integer*2	int
Name	Character*32	char*
PCASID	Character*32	char*
YCoef(moles/mole)	Real*8	Double *

3.2.21 Subroutine ChemAnaRed

This subroutine returns the anaerobic reduction rate constant (Rate) in units of 1/day and number of reaction products (NumProd).

FORTRAN Interface

Subroutine ChemAnaRed(Rate,NumProd)

C++ Prototype

void ChemAnaRed(Rate,NumProd)

Arguments	FORTRAN Type	C++ Type
Rate (1/day)	Real*8	Double *
NumProd	Integer*2	int *

3.2.22 Subroutine ChemPAnaRed

This subroutine returns data for the indexed product that is associated with the anaerobic reduction rate constant. The Index parameter defines which of the NumProd chemicals the rate is associated with. The name (Name), product CASID (PCASID), and molar yield coefficient for reaction (YCoef) in moles/mole are returned.

FORTRAN Interface

Subroutine ChemPAnaRed(Index,Name,PCASID,YCoef)

C++ Prototype

void ChemPAnaRed(Index,Name,PCASID,YCoef)

Arguments	FORTRAN Type	C++ Type
Index	Integer*2	int
Name	Character*32	char*
PCASID	Character*32	char*
YCoef(moles/mole)	Real*8	Double *

3.2.23 Subroutine ChemAnaBio

This subroutine returns the anaerobic biodegradation rate constant (Rate) in units of 1/day and number of reaction products (NumProd). The determination of acid, neutral, or base conditions is made from the value set for pH (see description of Subroutine ChemEnv, Section 3.1.1).

FORTRAN Interface

Subroutine ChemAnaBio(Rate,NumProd)

C++ Prototype

void ChemAnaBio(Rate,NumProd)

Argument	FORTRAN Type	C++ Type
Rate (1/day)	Real*8	Double *
NumProd	Integer*2	int *

3.2.24 Subroutine ChemPAnaBio

This subroutine returns data for the indexed product that is associated with the anaerobic biodegradation rate constant. The Index parameter defines which of the NumProd chemicals the rate is associated with. The name (Name), product CASID (PCASID), and molar yield coefficient for reaction (YCoef) in moles/mole are returned. The determination of acid, neutral, or base conditions is made from the value set for pH (see description of Subroutine ChemEnv, Section 3.1.1).

FORTRAN Interface

Subroutine ChemPAnaBio(Index,Name,PCASID,YCoef)

C++ Prototype

void ChemPAnaBio(Index,Name,PCASID,YCoef)

Arguments	FORTRAN Type	C++ Type
Index	Integer*2	int
Name	Character*32	char*
PCASID	Character*32	char*
YCoef(moles/mole)	Real*8	Double *

3.2.25 Subroutine ChemSO4Bio

This subroutine returns the SO_4 reducing biodegradation rate constant (Rate) in units of 1/day and number of reaction products (NumProd). The determination of acid, neutral, or base conditions is made from the value set for pH (see description of Subroutine ChemEnv, Section 3.1.1).

FORTRAN Interface

Subroutine ChemSO4Bio(Rate,NumProd)

C++ Prototype

void ChemSO4Bio(Rate,NumProd)

Argument	FORTRAN Type	C++ Type
Rate (1/day)	Real*8	Double *
NumProd	Integer*2	int *

3.2.26 Subroutine ChemPSO4Bio

This subroutine returns data for the indexed product that is associated with the SO_4 reducing biodegradation rate constant. The Index parameter defines which of the NumProd chemicals the rate is associated with. The name (Name), product CASID (PCASID), and molar yield coefficient for reaction (YCoef) in moles/mole are returned. The determination of acid, neutral, or base conditions is made from the value set for pH (see description of Subroutine ChemEnv, Section 3.1.1).

FORTRAN Interface

Subroutine ChemPSO4Bio(Index,Name,PCASID,YCoef)

C++ Prototype

void ChemPSO4Bio(Index,Name,PCASID,YCoef)

Arguments	FORTRAN Type	C++ Type
Index	Integer*2	int
Name	Character*32	char*
PCASID	Character*32	char*
YCoef(moles/mole)	Real*8	Double *

3.2.27 Subroutine ChemMetBio

This subroutine returns the methanogenic biodegradation rate constant (Rate) in units of 1/day and number of reaction products (NumProd). The determination of acid, neutral, or base conditions is made from the value set for pH (see description of Subroutine ChemEnv, Section 3.1.1).

FORTRAN Interface

Subroutine ChemMetBio(Rate,NumProd)

C++ Prototype

void ChemMetBio(Rate,NumProd)

Argument	FORTRAN Type	C++ Type
Rate (1/day)	Real*8	Double *
NumProd	Integer*2	int *

3.2.28 Subroutine ChemPMetBio

This subroutine returns data about a particular product that is associated with the methanogenic biodegradation rate constant. The name (Name), product CASID (PCASID), and molar yield coefficient for reaction (YCoef) in moles/mole are returned. The determination of acid, neutral, or base conditions is made from the value set for pH (see description of Subroutine ChemEnv, Section 3.1.1).

FORTRAN Interface
Subroutine ChemPMetBio(Index,Name,PCASID,YCoef)
C++ Prototype
void ChemPMetBio(Index,Name,PCASID,YCoef)

Arguments	FORTRAN Type	C++ Type
Index	Integer*2	int
Name	Character*32	char*
PCASID	Character*32	char*
YCoef(moles/mole)	Real*8	Double *

3.2.29 Function ChemHuman

This function returns a flag indicating whether human health benchmarks exist for the given CASID in the CPP HHB data table.

FORTRAN Interface
Function ChemHuman() Logical
C++ Prototype
int ChemHuman()

3.2.30 Function ChemRfDfood

This function returns the reference dose in mg/kg-d for food ingestion including the given CASID.

FORTRAN Interface
Function ChemRFD() Real*8
C++ Prototype
double ChemRFD()

3.2.31 Function ChemRfC

This function returns the reference concentration in mg/m^3 for inhalation including the given CASID.

FORTRAN Interface
Function ChemRFC() Real*8
C++ Prototype
double ChemRFC()

3.2.32 Function ChemRfDwater

This function returns the reference dose in mg/kg-d for water ingestion including the given CASID.

FORTRAN Interface
Function ChemRFDwater() Real*8
C++ Prototype
double ChemRFDwater()

3.2.33 Function ChemRfDfish

This function returns the reference dose in mg/kg-d for fish ingestion including the given CASID.

FORTRAN Interface
Function ChemRFDfish() Real*8
C++ Prototype
double ChemRFDfish()

3.2.34 Function ChemBreastMilkExp

This function returns a flag indicating whether breast milk is impacted by the given CASID.

FORTRAN Interface
Function ChemCSFwaterl() Integer*2
C++ Prototype
int ChemCSFwaterl()

3.2.35 Function ChemBM

This function returns the reference dose in mg/kg-d for oral intake of breast milk for the given CASID.

FORTRAN Interface
Function ChemBM() Real*8
C++ Prototype
double ChemBM()

3.2.36 Function ChemHealthEffect

This function returns the health effect of the given CASID.

FORTRAN Interface
Function ChemHealthEffect() Integer*2
C++ Prototype
int ChemHealthEffect()

3.2.37 Function ChemNC_Add

This function returns a flag indicating whether the ingestion and inhalation health effects can be added for the given CASID.

FORTRAN Interface
Function ChemNC_Add() Logical
C++ Prototype
int ChemNC_Add()

3.2.38 Function ChemCSFfood

This function returns ingestion Cancer Slope Factor for food ingestion in 1/(mg/kg-d) for the given CASID.

FORTRAN Interface
Function ChemCSFOral() Real*8
C++ Prototype
double ChemCSFOral()

3.2.39 Function ChemCSFinhal

This function returns inhalation Cancer Slope Factor for inhalation in 1/(mg/kg-d) of the given CASID.

FORTRAN Interface
Function ChemInhCSFl() Real*8
C++ Prototype
double ChemInhCSF()

3.2.40 Function ChemCSFwater

This function returns ingestion Cancer Slope Factor for oral intake of water in 1/(mg/kg-d) for the given CASID.

FORTRAN Interface
Function ChemCSFwaterl() Real*8
C++ Prototype
double ChemCSFwaterl()

3.2.41 Function ChemC_Add

This function returns a flag indicating whether the cancer incidence of ingestion and inhalation can be added for the given CASID.

FORTRAN Interface
Function ChemC_Add() Logical
C++ Prototype
int ChemC_Add()

3.2.42 Functions Chemfai, ChemFam, ChemFbl, and ChemFf

These functions contain fractions associated with breast milk intake for the given CASID.

FORTRAN Interface
Function Chemfai() Real*8
Function ChemFam() Real*8
Function ChemFbl() Real*8
Function ChemFf() Real*8
C++ Prototype
double Chemfai()
double ChemFam()
double ChemFbl()
double ChemFf()

3.2.43 Functions Chemkpm and ChemKrbc

These functions are parameters (unitless) associated with breast milk intake for the given CASID. Chemkpm is the concentration proportionality constant between plasma and breast milk in the aqueous phase; ChemKrbc is the concentration proportionality constant between red blood cells and plasma.

FORTRAN Interface
Function Chemkpm() Real*8
Function ChemKrbc() Real*8

C++ Prototype
double Chemkpm()
double ChemKrbc()

3.2.44 Function Chemt_halfb

This function returns the half time in days of the chemical in breast milk intake for the given CASID.

FORTRAN Interface
Function Chemt_halfb() Real*8
C++ Prototype
double Chemt_halfb()

3.2.45 Function ChemEco

This function will return a flag indicating whether ecological benchmarks exist for the selected chemical in the data tables that are used by the CPP.

FORTRAN Interface
Function ChemEco() Logical
C++ Prototype
int ChemEco()

3.2.46 Function ChemEB

This function will return the ecological benchmarks for all species. If the variable species does not contain a recognized species, the function will return .false. in FORTRAN and 0 in C/C++.

FORTRAN Interface
Function ChemEB(Species,Sed,Soil,WaterDis,WaterTot,EBRec) Logical
C++ Prototype
int ChemEB(Species,Sed,Soil,WaterDis,WaterTot,EBRec)

Argument	**FORTRAN Type**	**C++ Type**
Species	Character*32	char*
Sed (µg/g)	Real*8	double *
Soil (µg/g)	Real*8	double *
WaterDis(mg/L)	Real*8	double *
WaterTot(mg/L)	Real*8	double *
EBRec (mg/kg-day)	Real*8	double *

3.2.47 Function ChemSoilTo

This function returns the biological concentration factor for soil to the defined species. If the variable Species does not contain a recognized species, the function will return .false. in FORTRAN and 0 in C/C++.

FORTRAN Interface
Function ChemSoilTo(Species,BCF) Boolean
C++ Prototype
int ChemSoilTo(Species,BCF)

Argument	FORTRAN Type	C++ Type
Species (selected from "exveg," "proveg," "exfruit," "profruit," "root," "grain," "silage," "forage," "earthworm," "inverterbrate," "small mammal," and "other vertebrate")	Character*16	char*
BCF([µg/g DW species]/[µg/g soil])	Real*8	double*

3.2.48 Function ChemAirTo

This function returns the biological transfer factor for air to the defined plant. If the variable Plant does not contain a recognized plant, the function will return .false. in FORTRAN and 0 in C/C++.

FORTRAN Interface
Function ChemAirTo(Plant,BTF) Boolean
C++ Prototype
int ChemAirTo(Plant,BTF)

Argument	FORTRAN Type	C++ Type
Plant (selected from"exveg," "exfruit," "silage," and "forage")	Character*16	char*
BTF([µg/g DW plant]/[µg/g air])	Real*8	double*

3.2.49 Function ChemRCF

This function returns the root concentration factor (unitless) for soil to the root.

FORTRAN Interface
Function -ChemRCF() Real*8
C++ Prototype
double ChemRCF()

3.2.50 Function ChemBs

This function will return the bioavailablility fraction for contaminated soil.

FORTRAN Interface
Function ChemBs() Real*8
C++ Prototype
double ChemBs()

3.2.51 Function ChemBTF

This function returns the biotransfer factor in d/g for the given medium. If the variable Medium does not contain a recognized medium, the function will return .false. in FORTRAN and 0 in C/C++.

FORTRAN Interface
Function ChemBTF(Medium,Ba)Boolean
C++ Prototype
int ChemBTF(Medium,Ba)

Argument	**FORTRAN Type**	**C++ Type**
Medium (selected from “milk,” “beef,” and “water”)	Character*32	char*
Ba(d/g)	Real*8	double*

3.2.53 Function ChemMT

This function returns the metabolic transformation rate in units of 1/day.

FORTRAN Interface
Function ChemMT() Real*8
C++ Prototype
double ChemMT()

3.2.54 Subroutine ChemBAF

This function will return the biotransfer factor for the given species. If the variable species does not contain a recognized species, the function will return .false. in FORTRAN and 0 in C/C++. These species are selected from “birds_sm, “herbiverts, “herp_sm,” “invert,” “mammals_sm,” “omniverts,” and “worms.”

FORTRAN Interface

Function ChemBAF(Species,Ba) Boolean

C++ Prototype

int ChemBAF(Species,Ba)

Argument	FORTRAN Type	C++ Type
Species (selected from “birds_sm,” “herbiverts,” “invert,” “mammals_sm,” “omniverts,” and “worms”)	Character*32	char*
Ba(unitless)	Real*8	double*

3.2.55 Subroutine ChemkpPar

This subroutine returns the biological accumulation and concentration factor for soil to the defined species. If the variable Species does not contain a recognized species, the function will return .false. in FORTRAN and 0 in C/C++. These species are selected from “exveg,” “exfruit,” “silage,” and “forage.”

FORTRAN Interface

Function ChemkpPar(Species,kpPar) Boolean

C++ Prototype

int ChemkpPar(Species,kpPar)

Argument	FORTRAN Type	C++ Type
Species (selected from “exveg,” “exfruit,” “silage,” and “forage”)	Character*32	char*
kpPar(1/y)	Real*8	double*

3.2.56 Subroutine ChemkpVap

This subroutine returns the biological accumulation and concentration factor for soil to the defined species. If the variable Species does not contain a recognized species, the function will return .false. in FORTRAN and 0 in C/C++. This species is selected from “exveg,” “exfruit,” “silage,” and “forage.”

FORTRAN Interface
Function ChemkpVap(Species,kpVap) Boolean
C++ Prototype
int ChemkpVap(Species,kpvap)

Argument	FORTRAN Type	C++ Type
Species (selected from "exveg," "exfruit, "silage," and "forage")	Character*32	char*
kpVap(1/y)	Real*8	double*

3.2.57 Function ChemecfPlant

This function returns the empirical correction factor for Bv:

FORTRAN Interface
Function ChemecfPlant() Real*8
C++ Prototype
double ChemecfPlant()

3.2.58 Functions ChemaqmpBCFm, ChembenthffBAFm, ChemT3fishBAFm, ChemT4fishBAFm, ChemT3musBAFm, and ChemT4musBAFm

These functions return bioconcentration factors. ChemaqmpBCFm returns the aquatic bioconcentration factor for plants in units of L/kg plant tissue. ChembenthffBAFm returns the aquatic bioconcentration factor for benthos in units of L/kg benthos. ChemT3fishBAFm returns the aquatic bioconcentration factor for T3 finfish in L/Kg. ChemT4fishBAFm returns the aquatic bioconcentration factor for T4 finfish in L/Kg. ChemT3musBAFm returns the aquatic bioconcentration factor for T3 fish tissue in L/Kg. ChemT4musBAFm returns the aquatic bioconcentration factor for T4 fish tissue in L/Kg.

FORTRAN Interface
Function ChemaqmpBCFm() Real*8
Function ChembenthffBAFm() Real*8
Function ChemT3fishBAFm() Real*8
Function ChemT4fishBAFm() Real*8
Function ChemT3musBAFm() Real*8
Function ChemT4musBAFm() Real*8

C++ Prototype
double ChemaqmpBCFm()
double ChembenthffBAFm()
double ChemT3fishBAFm()
double ChemT4fishBAFm()
double ChemT3musBAFm()
double ChemT4musBAFm()

3.2.59 Function NumNegIon and NumPosIon

These functions return the number of negative ion pKas reported and the number of positive ion pKbs for the given chemical. The return value is between 0 and 2 inclusive.

FORTRAN Interface
Function NumNegIon() Integer*4
Function NumPosIon() Integer*4
C++ Prototype
int NumNegIon()
int NumPosIon()

3.2.60 Function ChemNegIonSpecies and ChemPosIonSpecies

These functions return a flag stating whether the indexed ion species has an associated pKa or pKb. The number of negative ion pKas reported and the number of positive ion pKbs for the given chemical is given in functions NumNegIon and NumPosIon.

FORTRAN Interface
Function ChemNegIonSpecies(Ion) logical
Function ChemPosIonSpecies(Ion) logical
C++ Prototype
int ChemNegIonSpecies(Ion)
int ChemPosIonSpecies(Ion)

Argument	FORTRAN Type	C++ Type
Ion	Integer*4	int

3.2.61 Function ChemNegIonSpecies and ChemPosIonSpecies

These functions return the pKa or pKb for the indexed ion species. The number of negative ion pKas reported and the number of positive ion pKbs for the given chemical is given in functions NumNegIon and NumPosIon. This function returns its value in the same units as pH.

FORTRAN Interface

Function ChemNegIonpKa(Ion) logical

Function ChemPosIonpKb(Ion) logical

C++ Prototype

int ChemNegIonpKa(Ion)

int ChemPosIonpKb(Ion)

Argument	**FORTRAN Type**	**C++ Type**
Ion	Integer*4	int

3.2.62 Function ChemFracNeutral

This function returns the fraction neutral for the given chemical.

FORTRAN Interface

Function ChemFracNeutral() Real*8

C++ Prototype

double ChemFracNeutral()

4.0 Testing Approach and Results

This section describes the type of testing conducted for the CPP, summarizes the requirements on which testing was based, and describes test cases and results of their implementation. Additional information related to testing can be found in Appendix A.

Associated with this processor are several data tables stored in flat-ASCII format and populated by EPA: OCP, MICP, actalyzation (CAT), ActBio, AerBio, AnaBio, AnaRedBio, MethBio, SO4Bio, HHB, EB, EBF, ABF, Chemical Ecological Flag, and Waste Concentration.

4.1 Type of Testing

Software can be tested at both the unit and system levels. Unit testing evaluates individual components in isolation from other components (for example, the CPP in isolation from the FRAMES-HWIR Technology Software System). System testing evaluates the performance of groups of components functioning together, data communication between the components comprising the system (also called integration testing), and the overall performance of the system (for example, testing the functioning of the CPP within the FRAMES-HWIR Technology Software System). This test plan currently addresses unit testing only. Once additional pieces of the system have been completed (including their own unit testing), this test plan will be revised to include other tests cases needed to test the CPP within the system. For example, additional cases will also be designed to test the CPP's interface with the SUI, SDP, and MMSP.

Note that at the time of unit testing, data tables had not been fully populated by the EPA. Accordingly, surrogate data were used for unit testing. The OCP included data for 312 chemicals, and the MICP included data (some of it surrogate) for 8 chemicals, for a total of 320 chemicals. However, data in many of the other tables were associated with three chemicals only, one metal/inorganic (mercury) and two organics (acenaphthene and acetic acid). Thus, all test cases focus on those three chemicals.

4.2 Summary of Requirements

Requirements for the CPP are summarized in Section 2.0 of this document. These requirements were reworded into the list in Table 4.1. They were stated as concise, fundamental requirements that are suitable for testing.

To ensure that the CPP meets the requirements listed in Table 4.1, test cases were developed to check operation. Some of these test cases focus on evaluating whether the CPP can respond to each command individually or in groups. Individual calls to the CPP are only made by the Aerated Tank and Surface Impoundment Modules within the MMSP. All other components of the FRAMES-HWIR Technology Software System that interact with the CPP do so through the chemical properties site simulation file (CP.SSF). Table 4.2 shows the relationship between the all test cases and requirements.

Table 4.1 Fundamental Requirements for Testing the Chemical Properties Processor

Requirement Number	Requirement
1	Appropriately determine which data tables to read (both location and whether information on an organic or inorganic/metal is being requested).
2	Correctly compute pH and temperature-dependent organic chemical properties using mathematical relationships defined in the document "HWIR Chemical Database" by accessing the OCP.
3	Correctly compute certain chemical properties using a statistical random sampling of distributions in the MICP, CAT, ActBio, AerBio, AnaBio, AnaRedBio, MethBio, and SO4Bio data tables.
4	Return appropriate data when called from the ABF, CAT, EB, EBF, HHB, Chemical Ecological Flag, and Waste Concentration.
5	Read each of 15 data tables only, which are formatted in a specified flat-ASCII format and populated by EPA (OCP, MICP, CAT, ActBio, AerBio, AnaBio, AnaRedBio, MethBio, SO4Bio, HHB, EB, EBF, ABF, Chemical Ecological Flag, and Waste Concentration).
6	Write requested chemical information to a chemical properties data group for the SDP to use in creating the SSF.
7	Write an error message to the error file and halt program execution when a designated error condition occurs (e.g., CPP.DLL has not been initialized)
8	Produce a 0 when a parameter is not available in the database, and allow the calling module or processor to determine whether the condition warrants a warning or error (e.g., when data on a species is requested and that species is not in the database).
9	Support the Microsoft® Visual C++ Version 5.0 compiler.
10	Support the Borland® C++ Version 4.0 compiler.

4.3 Test Cases

The approach to meeting the CPP requirements was to design the processor as a DLL. DLLs are called by programs. Thus, to test the CPP, a testing program is used that performs subroutine and function calls to the DLL as specified in a test-specific input file, *testname*test.tst. For example, CPP_01Btest.tst would be the test-specific input file for Test Case CPP_01 for a Borland® C++ compiler, and CPP_01Mtest.tst would be the test-specific input file for Test Case CPP_01 for a Microsoft® C++ compiler.

All tests were conducted under Windows 95® because this is the only operating system supported. In addition, all tests were conducted with Borland C++ 4.0 and Microsoft Visual C++ 5.0 compiled versions of the testing program. Note that additional compilers such as Lahey FORTRAN-90 4.0 and Digital Visual FORTRAN-90 5.0 were not developed or tested because no components of the FRAMES-HWIR Technology Software System that would interact with the CPP needed them.

Table 4.2 Relationship Between Test Cases and Fundamental Requirements

		Test Case Name (CPP_xx)								
		01	02	03	04	05	06	07	08	09
Requirement	1	x	x	x	x	x	x	x	x	
	2		x	x		x				
	3		x	x	x	x	x			
	4					x	x			
	5					x	x			
	6					x	x			
	7							x	x	x
	8			x	x	x	x			
	9	x	x	x	x	x	x	x	x	x
	10	x	x	x	x	x	x	x	x	x

The general procedure for conducting each test is documented in Appendix A. For testing purposes, a full set of test cases for each compiler type was included in subfolders to the CPP Test Bed Directory. Therefore, each test case is run twice, once for each compiler. Test cases for the Microsoft® C++ compiler are in subdirectory mscpp; test cases for the Borland® C++ compiler are in the subdirectory bccpp.

Note that at the time of unit testing, few data were available to fill the data tables. Accordingly, surrogate data were used such that it was readily apparent if an incorrect answer was provided. For example, each animal in the EB data table was given a unique number so that differences in reporting were obvious. Also, statistical data were provided for chemicals only in the AerBio data table because the same routines are used regardless of which table the data are selected from.

4.3.1 CPP_01

4.3.1.1 Description and Rationale

For any of the chemical properties to be calculated or selected as appropriate, the CPP must be initialized. This test case evaluates the ability of the CPP to run the three initialization subroutines (ChemEnv, ChemCASID, and ChemPath) for an organic chemical. These subroutines must be run regardless of whether a system component is interacting through the SSF or calling the CPP individually. Subroutine ChemEnv sets the temperature, pH, and media (soil, sediment, or surface water) for the chemical information. ChemCASID provides a unique identification number for the chemical. ChemPath lists the locations of the various data tables that will be called by the other subroutines and functions.

4.3.1.2 Input Data

Input file CPP_01xtest.tst was used for the two C++ compilers (where "x" is either B for Borland or M for Microsoft). The calls it simulates are as follows:

ChemPath

..\CPPData
ChemEnv
20.0,7.0,Sediment,12
ChemCASID
71-43-2
Stop
Pause

In these calls, ChemPath provides the location of the data tables (in subdirectory CPPData within the CPP Test Bed Directory), ChemEnv calls that subroutine, 20.0 represents the temperature, 7.0 represents the pH, Sediment is the medium, 0.2 is the Foc value, ChemCASID calls that subroutine, 71-43-2 represents the identification number for the chemical benezene, Pause holds the information on the screen long enough for the tester to verify what was done, and Stop halts program execution.

4.3.1.3 Expected Results

It is expected that the CPP will use the data tables in the bccpp (or mscpp if appropriate) directory and that subroutines ChemEnv and ChemCASID will execute without error or without returning a 0. It is also expected that the CPP will produce a file CPP_01xtest.out (where "x" stands for B for Borland or M for Microsoft), which will contain the expected values, namely, a temperature value of 20, a pH value of 7.0, the medium as Sediment, an Foc of 0.2, and a CASID of 71-43-2.

4.3.1.4 Procedure

Using Windows® Explorer, the tester double-clicks on the file CPP_01B.bat in the bccpp directory or CPP_01M.bat in the mscpp. When the program finishes running, the tester compares the calls that were made to those that were requested. The tester then exits the DOS window where the calls were displayed and verifies that the CPP produced the file CPP_01xtest.out (where "x" stands for B for Borland or M for Microsoft). The tester then double-clicks on the output file and compares the values in it against what was expected (see above). The tester also checks the global results files (GRF) directory to make sure that no error or warning files were produced. Note that a warning file may be produced to verify the chemical being requested.

4.3.1.5 Results

The program executed as expected. Values produced in the output file matched those that were expected. No error file was produced; a warning file was produced as expected to verify the chemical being requested. Therefore, the CPP passed this test case.

4.3.2 CPP_02

4.3.2.1 Description and Rationale

As mentioned, the CPP must randomly sample from the ActBio, AerBio, AnaBio, MethBio, and SO4Bio data tables. This test case evaluates the ability of the CPP to randomly sample inorganic chemical data in the correct distribution type. Because data across all 15 tables were not available consistently, surrogate data were used for seven chemicals to allow generation of all distribution types.

The distribution types are normal, log normal, exponential, uniform, Johnson SB, Johnson SU, DEmp, and triangular (see TetraTech 1998).

4.3.2.2 Input Data

Input file CPP_02xtest.tst was used for the two C++ compilers (where "x" is either B for Borland or M for Microsoft). The calls it simulates are as follows:

```
ChemPath
..\CPPData
ChemEnv
20.0,7.0,Sediment,0.2
OpenGroups
ChemCASID, ; Checking  Different Distributions through the AerBio.csv
7439-97-6
ChemAerBio
ChemAerBio
ChemAerBio
ChemAerBio
ChemCASID
71-43-2,
ChemAerBio
ChemAerBio
ChemAerBio
ChemAerBio
ChemCASID
108-95-2,
ChemAerBio
ChemAerBio
ChemAerBio
ChemAerBio
ChemCASID
108-88-3,
ChemAerBio
ChemAerBio
ChemAerBio
ChemAerBio
ChemCASID
7440-22-4,
ChemAerBio
ChemAerBio
ChemAerBio
ChemAerBio
ChemCASID
16065-83-1,
ChemAerBio
ChemAerBio
```

ChemAerBio
ChemAerBio
ChemCASID
50-32-8,
ChemAerBio
ChemAerBio
ChemAerBio
ChemAerBio
ChemCASID
71-43-2,
ChemMetBio
ChemMetBio
ChemMetBio
ChemMetBio
ChemMetBio
ChemMetBio
ChemMetBio
ChemMetBio
ChemMetBio
ChemMetBio
ChemMetBio
ChemMetBio
ChemMetBio
ChemMetBio
ChemMetBio
ChemMetBio
ChemMetBio
ChemMetBio
ChemMetBio
ChemMetBio
CloseGroups
Stop
Pause

In these calls, each of the items starting with Chem calls that subroutine. ChemAerBio is called four times to ensure that a random number is being generated. ChemMetBio is called specifically to test the DEmp distribution type. This distribution type did not appear in the AerBio table. The rest of the calls are described in Section 4.3.1.2.

4.3.2.3 Expected Results

It is expected that all subroutines and functions called will execute without error or without returning a 0. It is also expected that the CPP will produce file CPP_02xtest.out (where "x" stands for B for Borland or M for Microsoft), which will contain the expected values, namely a temperature value of 20, a pH value of 7.0, the medium as sediment, an Foc of 0.2, and the appropriate CASIDs. A warning file will also be produced that verifies the distribution types used. The distribution type and range for each chemical is shown in Table 4.3. These values are taken from *Design and Test Plan for Random Sampling Subroutines* (TetraTech 1998).

4.3.2.4 Procedure

Using the Windows® Explorer, the tester double-clicks on the file CPP_02x.bat (where "x" stands for B for Borland or M for Microsoft). When the program finishes running, the tester compares the calls that were made to those that were requested by checking the warning file in the GRF directory. The tester then verifies that the CPP produced the file CPP_02xtest.out. The tester then double-clicks on the output file and compares the values in it against what was expected (see Table 4.3), making sure that each of the four values for each parameter differ from each other and fall with the minimum and maximums allowed. The tester also checks the GRF directory to ensure that no error file was produced.

4.3.2.5 Results

The program executed as expected. Values produced by the output file were within the range of those that were expected. No warning or errors files were produced, except the warning file listing the distribution types. Therefore, the CPP passed this test case.

Table 4.3 Distribution Types and Ranges Expected for Chemicals

CASID	Distribution Type	Mean	Minimum	Maximum
7439-97-6	Normal	7.62	1	25.4
7440-22-4	Log-Normal	45.9	2.1	200
16065-83-1	Exponential	7.62	1	25.4
108-95-2	Uniform	NA	0.024	11
71-43-2	Triangular	264	112	560
108-88-3	Johnson SB	45.9	2.1	200
50-32-8	Johnson SU	0.00118	0	0.00235
71-43-2	DEmp	one of 0, 0, 0, 0, 0, 0.005, 0.0074		

4.3.3 CPP_03

4.3.3.1 Description and Rationale

One of the primary functions of the CPP is to provide consistent chemical data for two of the modules within the MMSP (the aerated tank and surface-impoundment source modules). The CPP accomplishes this by pulling data from a series of data tables, beginning with either the OCP or the MICP. This test case evaluates the ability of the CPP to pull data on two organic chemicals and provide these data in a file for use by the modules.

4.3.3.2 Input Data

Input file CPP_03xtest.tst was used for the two C++ compilers (where "x" is either B for Borland or M for Microsoft). The calls it simulates are as follows:

OpenGroups
ChemPath
..\CPPData
ChemEnv
20.0,7.0,Sediment,0.2
ChemCASID
108-95-2
ChemSMILES
ChemMolWt, ; expected result 94.11
ChemADiff, ; expected result .0768
ChemVol, ; expected result 89.7
ChemDen, ; expected result 1.05
ChemWDiff, ; expected result 8.94E-6
ChemVP, ; expected result 2.83E-01
ChemSol, ; expected result 4.86E+04
ChemHLC, ; expected result 7.67E-07
ChemKow, ; expected result 38.1
ChemKoc, ; expected result 18.3
ChemKd, ; expected result 3.66
ChemCASID
78-93-3
ChemSMILES
ChemMolWt, ; expected result 72.11
ChemADiff, ; expected result .0876
ChemVol, ; expected result 89.6
ChemDen, ; expected result 0.805
ChemWDiff, ; expected result 8.95E-06
ChemVP, ; expected result 64
ChemSol, ; expected result 1.42E+05
ChemHLC, ; expected result 4.62
ChemKow, ; expected result 2.33
ChemKoc, ; expected result 1.11
ChemKd, ; expected result 0.22
ChemCASID
108-95-2,
ChemAerBio
ChemAerBio
ChemAerBio
ChemAerBio
ChemCASID
78-93-3,
ChemAerBio
ChemAerBio
ChemAerBio
ChemAerBio
ChemCASID, ; Checking Transformation products information
75-15-0

ChemHyd
ChemPHyd
0
ChemAerBio
ChemPAerBio
0
ChemPAerBio
2
ChemActBio
ChemPActBio
0
ChemPActBio
2
ChemAnaRed
ChemPAnaRed
0
ChemPAnaRed
2
ChemAnaBio
ChemPAnaBio
0
ChemPAnaBio
2
ChemSO4Bio
ChemPSO4Bio
0
ChemPSO4Bio
2
ChemMetBio
ChemPMetBio
0
ChemPMetBio
2
ChemCASID , ;Checking Transformation products information
78-93-3
ChemHyd
ChemPHyd
0
ChemPHyd
8
ChemAerBio
ChemPAerBio
0
ChemPAerBio
2
ChemActBio
ChemPActBio

0
ChemPActBio
2
ChemAnaRed
ChemPAnaRed
0
ChemPAnaRed
2
ChemAnaBio
ChemPAnaBio
0
ChemPAnaBio
2
ChemSO4Bio
ChemPSO4Bio
0
ChemPSO4Bio
2
ChemMetBio
ChemPMetBio
0
ChemPMetBio
2
CloseGroups
Stop

In these calls, each of the items starting with Chem calls that subroutine. ChemAerBio is called four times to ensure that a random number is being generated. This random set of numbers demonstrates that the number is being generated within the distribution given in the HWIR chemical database. The rest of the calls are described in Section 4.3.1.2.

4.3.3.3 Expected Results

It is expected that all subroutines and functions called will execute without error or without returning a 0. It is also expected that the CPP will produce file CPP_03xtest.out (where "x" stands for B for Borland or M for Microsoft), which will contain the expected values, namely, a temperature value of 20, a pH value of 7.0, the medium as sediment, an Foc of 0.2, the appropriate CASIDs, and the expected results listed in Section 4.3.3.2. Those parameters without listed results would generate random numbers or numbers that differ in reality from the test case.

4.3.3.4 Procedures

Using the Windows® Explorer, the tester double-clicks on the file CPP_03x.bat (where "x" stands for B for Borland or M for Microsoft). When the program finishes running, the tester compares the calls that were made to those that were requested by reviewing the calls on the screen. The tester then verifies that the CPP produced the file CPP_03xtest.out. The tester then double-clicks on the output

file and compares the values in it against what was expected. The tester also checks the GRF directory to ensure that no error or warning files were produced other than the warning file verifying the commands.

4.3.3.5 Results

The program executed as expected. Values produced by the output file were within the range of those that were expected. No warning or errors files were produced other than the warning file verifying the commands. Therefore, the CPP passed this test case.

4.3.4 CPP_04

4.3.4.1 Description and Rationale

As mentioned, one of the primary functions of the CPP is to provide consistent chemical data for two of the modules within the MMSP (the aerated tank and surface impoundment source modules). The CPP accomplishes this by pulling data from a series of data tables, beginning with either the OCP or the MICP. This test case evaluates the ability of the CPP to pull data on three inorganic chemicals and provide these data in a file for use by the modules. The file generated is a chemical-properties site simulation file.

4.3.4.2 Input Data

Input file CPP_04xtest.tst was used for the two C++ compilers (where "x" is either B for Borland and M for Microsoft). The calls it simulates are as follows:

```
OpenGroups
ChemPath
..\CPPData
ChemEnv
20.0,7.0,Sediment,0.2
ChemCASID
7439-97-6
ChemSol
ChemMolWt
ChemKd
ChemCASID
7440-22-4
ChemSol
ChemMolWt
ChemKd
ChemCASID
16065-83-1
ChemSol
ChemMolWt
ChemKd
CloseGroups
```

Stop
Pause

In these calls, each of the items starting with Chem calls that subroutine. ChemAerBio is called four times to ensure that a random number is being generated. The rest of the calls are described in Section 4.3.1.2.

4.3.4.3 Expected Results

It is expected that all subroutines and functions called will execute without error or without returning a 0. It is also expected that the CPP will produce file CPP_04xtest.out (where "x" stands for B for Borland or M for Microsoft), which will contain the expected values, namely a temperature value of 20, a pH value of 7.0, the medium as Soil, an Foc of 0.2, and the appropriate CASID.

4.3.4.4 Procedures

Using the Windows® Explorer, the tester double-clicks on the file CPP_04x.bat (where "x" stands for B for Borland or M for Microsoft). When the program finishes running, the tester compares the calls that were made to those that were requested by reviewing the calls on the screen. The tester then verifies that the CPP produced the file CPP_04xtest.out. The tester then double-clicks on the output file and compares the values in it against what was expected. The tester also checks the GRF directory to ensure that no error or warning files were produced.

4.3.4.5 Results

The program executed as expected. Values produced by the output file were within the range of those that were expected. No warning or errors files were produced other than the warning file verifying the commands. Therefore, the CPP passed this test case.

4.3.5 CPP_05

4.3.5.1 Description and Rationale

Another of the primary functions of the CPP is to produce a chemical property site simulation file (CP.SSF) for the SDP, using either the organic or metals/inorganic chemical data. This test case evaluates the ability of the CPP to produce a chemical SSF using the OCP.

4.3.5.2 Input Data

Input file CPP_05xtest.tst was used for the two C++ compilers (where "x" is either B for Borland and M for Microsoft). The calls it simulates are as follows:

OpenGroups
ChemPath
..\CPPData
ChemEnv
20.0,7.0,Sediment,0.2
AddGroup

cp1.ssf
ChemCASID
108-88-3
ChemSSF
cp1.ssf
RemoveGroup
cp1.ssf
AddGroup
cp2.ssf
ChemCASID
108-95-2
ChemSSF
cp2.ssf
RemoveGroup
cp2.ssf
AddGroup
cp3.ssf
ChemCASID
75-01-4
ChemSSF
cp3.ssf
RemoveGroup
cp3.ssf
AddGroup
cp4.ssf
ChemCASID
67-66-3
ChemSSF
cp4.ssf
RemoveGroup
cp4.ssf
CloseGroups
Stop
Pause

In these calls, each of the items starting with Chem calls that subroutine. ChemAerBio is called four times to ensure that a random number is being generated. The rest of the calls are described in Section 4.3.1.2.

4.3.5.3 Expected Results

It is expected that all subroutines and functions called will execute without error or without returning a 0. It is also expected that the CPP will produce file CPP_05xtest.out (where "x" stands for B for Borland or M for Microsoft), which will contain the expected values, namely a temperature value of 20, a pH value of 7.0, the medium as Sediment, an Foc of 0.2, and the appropriate CASIDs. In addition, it is expected that the CPP will produce in the SSF directory files entitled cp1.ssf, cp2.ssf, cp3.ssf, and cp4.ssf, which should contain everything needed to provide input to the CP.SSF for the SDP.

4.3.5.4 Procedures

Using the Windows® Explorer, the tester double-clicks on the file CPP_05x.bat (where "x" stands for B for Borland or M for Microsoft). When the program finishes running, the tester compares the calls that were made to those that were requested by reviewing the calls on the screen. The tester then verifies that the CPP produced the file CPP_05xtest.out. The tester then double-clicks on the output file and compares the values in it against what was expected. The tester also checks the SSF directory to ensure that the four cp.ssf files were produced and contain the correct values. The tester also checks the GRF directory to ensure that no error or warning files were produced.

4.3.5.5 Results

The program executed as expected. Values produced by the output file were within the range of those that were expected. No warning or errors files were produced other than the warning file verifying the commands. Therefore, the CPP passed this test case.

4.3.6 CPP_06

4.3.6.1 Description and Rationale

As mentioned, another of the primary functions of the CPP is to produce a CP.SSF for the SDP, using either the organic or metals/inorganic chemical data. This test case evaluates the ability of the CPP to produce a chemical SSF using the MICP.

4.3.6.2 Input

Input file CPP_06xtest.tst was used for the two C++ compilers (where "x" is either B for Borland and M for Microsoft). The calls it simulates are as follows:

```
OpenGroups
ChemPath
..\CPPData
ChemEnv
20.0,7.0,Sediment,0.2
AddGroup
cp1.ssf
ChemCASID
7440-38-2
ChemSSF
cp1.ssf
RemoveGroup
cp1.ssf
AddGroup
cp2.ssf
ChemCASID
7440-22-4
ChemSSF
```

cp2.ssf
RemoveGroup
cp2.ssf
AddGroup
cp3.ssf
ChemCASID
16065-83-1
ChemSSF
cp3.ssf
RemoveGroup
cp3.ssf
AddGroup
cp4.ssf
ChemCASID
7440-66-6
ChemSSF
cp4.ssf
RemoveGroup
cp4.ssf
CloseGroups
Stop
Pause

In these calls, each of the items starting with Chem calls that subroutine. ChemAerBio is called four times to ensure that a random number is being generated. The rest of the calls are described in Section 4.3.1.2.

4.3.6.3 Expected Results

It is expected that all subroutines and functions called will execute without error or without returning a 0. It is also expected that the CPP will produce file CPP_06xtest.out (where "x" stands for B for Borland or M for Microsoft), which will contain the expected values, namely a temperature value of 20, a pH value of 7.0, the medium as sediment, an Foc of 0.2, and the appropriate CASIDs. In addition, it is expected that the CPP will produce in the SSF directory the four cp.ssf files, which should contain everything needed to provide input to the chemical properties site simulation file for the SDP.

4.3.6.4 Procedures

Using the Windows® Explorer, the tester double-clicks on the file CPP_06x.bat (where "x" stands for B for Borland or M for Microsoft). When the program finishes running, the tester compares the calls that were made to those that were requested by reviewing the calls on the screen. The tester then verifies that the CPP produced the file CPP_06xtest.out. The tester then double-clicks on the output file and compares the values in it against what was expected. The tester also checks the SSF directory to ensure that the four cp.ssf files were produced and contain the correct values. The tester also checks the GRF directory to ensure that no error or warning files were produced.

4.3.6.5 Results

The program executed as expected. Values produced by the output file were within the range of those that were expected. No warning or errors files were produced other than the warning file verifying the commands. Therefore, the CPP passed this test case.

4.3.7 CPP_07

4.3.7.1 Description and Rationale

The CPP must be able to report its status reliably to the SUI through the use of warning and error messages. This test case evaluates the CPP's ability to recognize a potential problem, in this case the provision of an incorrect location for the data tables, and to produce an error file to relay information to the SUI.

4.3.7.2 Input Data

This test case uses input file CPP_07x.test (where "x" stands for B for Borland or M for Microsoft), which contains all the commands necessary to access all data tables and produce results for all parameters in two cp.ssf files. The second command, ChemPath, lists the location of the data tables as a directory called Bogus, which does not exist.

4.3.7.3 Expected Results

The CPP should attempt to execute and stop when data tables are not found in the directory given. An error file should be produced in the GRF directory.

4.3.7.4 Procedures

The tester double-clicks on the file CPP_07x.bat. When the processor stops, the tester checks to see that no information is contained in the output file. The tester then checks the GRF directory to ensure that an error file was produced and that it correctly diagnoses the problem.

4.3.7.5 Results

The CPP stopped after the first CASID was called. The output file contained only verification that the CPP had reached that point in processing. The following error message was produced:

"Failed to call CloseGroups",
"Chemical missing in OCP.csv or MICP.csv datatable. CASID in Warning File",

The CASID in the warning file matched the one called for in the input file. Therefore, the CPP passed this test case.

4.3.8 CPP_08

4.3.8.1 Description and Rationale

The CPP must be able to recognize conditions that would provide a 0 (in C++) result from the database. This type of condition could result in a warning from the module or processor that called for the information and potentially inappropriate data.. This test case evaluates the CPP's ability to recognize another of the foreseen conditions: requesting an unrecognized CASID.

4.3.8.2 Input Data

This test case uses input file CPP_08x.test (where "x" stands for B for Borland or M for Microsoft), which contains all the commands necessary to access all data tables and produce results for all parameters in two cp.ssf files. The CASID requested, 9100-01-1, did not exist in the data tables during unit testing.

4.3.8.3 Expected Results

It is expected that the CPP will stop processing when it reaches the incorrect CASID and that an error and warning file will be produced that correctly diagnose the problem.

4.3.8.4 Procedures

Using Windows Explorer, the tester double-clicks on the file CPP_08x.bat. When the program quits processing, the tester exits the DOS window where the calls were displayed and verifies that the CPP produced the file CPP_08xtest.out. The tester then checks that error and warning files were produced with the correct identification of the problem.

4.3.8.5 Results

The CPP stopped after the first CASID was called. The output file contained only verification that the CPP had reached that point in processing. The following error message was produced:

"Failed to call CloseGroups",
"Chemical missing in OCP.csv or MICP.csv datatable. CASID in Warning File",

The CASID in the warning file matched the one called for in the input file. Therefore, the CPP passed this test case.

4.3.9 CPP_09

4.3.9.1 Description and Rationale

As mentioned, the CPP must also be able to recognize errors and terminate processing. This test case evaluates the CPP's ability to recognize a foreseen error: the lack of initialization of the CPP.DLL. For all calculations, the two initialization programs must be called first. Failure to initialize the CPP.DLL will result in an error.

4.3.9.2 Input Data

Input file CPP_09xtest.tst was used for the two C++ compilers (where "x" equals B for Borland and M for Microsoft). Although it contained several miscellaneous calls, it was missing the CASID and ChemEnv parameter information.

4.3.9.3 Expected Results

It is expected that the CPP will stop processing when it attempts to initialize and that an error and warning file will be produced that correctly diagnose the problem.

4.3.9.4 Procedures

Using Windows® Explorer, the tester double-clicks on the file CPP_09xtest.bat. When the program quits processing, the tester exits the DOS window where the calls were displayed and verifies that the CPP produced the file CPP_09xtest.out. The tester then checks that error and warning files were produced with the correct identification of the problem.

4.3.9.5 Results

The CPP stopped after the first few commands were passed. The output file contained only verification that the CPP had reached that point in processing. The following error message was produced:

"Failed to call CloseGroups",
"ChemEnv or ChemCASID must be called before ChemSSF",

Therefore, the CPP passed this test case.

4.4 Verification Testing

In addition to the tests performed for unit testing, two verification tests were performed on request from the EPA. These tests were performed with all components of the system, not just the CPP. The following protocol was used:

Step 1: Check that all temperature- and pH-independent transfers from tables to SSFs were properly performed.

The CPP logic for many parameters simply transfers information from a table to the SSF. As only two chemicals are used in the verification process, these checks only need to be performed for each chemical. This check is performed by visually looking at the data in the tables and comparing it to the data in the prints of the SSF. The specific parameters to check are too numerous to list here but all these parameters are simply transferred from the table to the chemical properties SSF.

- Step 1.1: Check ABF parameters transfer to CPAq.ssf for both sites--no errors found
- Step 1.2: Check Chemical Ecological Flag data table parameters transfer to CPAq.ssf for both sites--no errors found

- Step 1.3: Check concentrations in the Waste Concentration data table parameters transfer to CPAq.ssf for both sites—no errors found

- Step 1.4: Check EB data table parameters transfer to CPAq.ssf for both sites—the tester noted that the EB.CSV data table appears to be missing a value for benzene. The CPP behaved as expected. EPA was alerted that the data table is probably in error.

- Step 1.5: Check EBF data table parameters transfer to CPAq.ssf for both sites—no errors found

- Step 1.6: Check HHB data table parameters transfer to CPAq.ssf for both sites—no errors found

Step 2: Check that temperature- and pH-dependent distributions were properly sampled.

A calculation of this nature is never performed for the two selected chemicals for the verification tests (tests were selected by EPA). Given that there is no information for these two chemicals, the following parameters should be 0: ChemActBioNumProd, ChemActBioRate, ChemAerBioNumProd, ChemAerBioRate, ChemAnaBioNumProd, ChemAnaBioRate, ChemAnaRedNumProd, ChemAnaRedRate, ChemHydNumProd, ChemHydRate, ChemMetBioNumProd, ChemMetBioRate, ChemSO4BioNumProd, and ChemSO4BioRate. These are the biodegradation and reduction rates for the chemicals. The tester found that all of the above parameters were 0.

Step 3: Check that temperature- and pH-depedent calculations were properly performed.

The CPP logic for OCP and CAT data tables is based on pH and temperature. A set of 20 pH and temperature combinations covers all the media at the two sites to be verified. The OCP and CAT computations were verified by spreadsheet computations. Two spreadsheets performed these computations for both benzene and 2,3,7,8 TCDD. The specific chemical properties to be checked were ChemADiff, ChemDen, Chem HLC, ChemHydNumProd, ChemHydRate, ChemKd, ChemKoc, ChemKow, ChemSol, ChemVol, ChemVP, and ChemWDiff.

- Step 3.1: Check that all OCP calculations are performed properly and transferred to all CP.SSF for benzene--ChemSol and ChemKd for benzene were incorrect. An error in the CPP was found and corrected. All other parameters matched spreadsheet calculations.

- Step 3.2: Check that all OCP calculations are performed properly and transferred to all the CP.SSF for 2,3,7,8 TCDD--ChemSol and ChemKd for 2,3,7,8 TCDD were incorrect. An error in the CPP was found and corrected. All other parameters matched spreadsheet calculations.

5.0 Quality Assurance Program

The CPP was developed under a quality assurance program documented in Gelston et al. (1998). That program defines quality as the ability of the software to meet client needs. Meeting client needs starts with a shared understanding of how the software must perform and continues throughout the software life cycle of design, development, testing, and implementation through attention to details.

Figure 5.1 outlines the software development process used for the CPP, highlighting the quality check points. (Note: the CPP activities flow down the left side of the figure, because it is software developed for the first time, as opposed to a modification to existing software.) The process shown is designed for compatibility with similar processes used by other government agencies. For example, this quality process compares favorably with that in the EPA Directive 2182, *System Design and Development Guidance* (EPA 1997). It also compares favorably with the Office of Civilian Radioactive Waste Management's *Quality Assurance Requirements and Description, Supplement I, Software* (OCRWM 1995). Activities roughly equivalent across these processes are shown in Table 5.1.

Development of the CPP included the implementation of a quality assurance checklist (see Figure 5.2). Understanding of this checklist by all team members resulted in the shared understanding of component requirements and design necessary to ensure quality. Completion of this checklist verified that all documentation was completed for transfer of the software to client use.

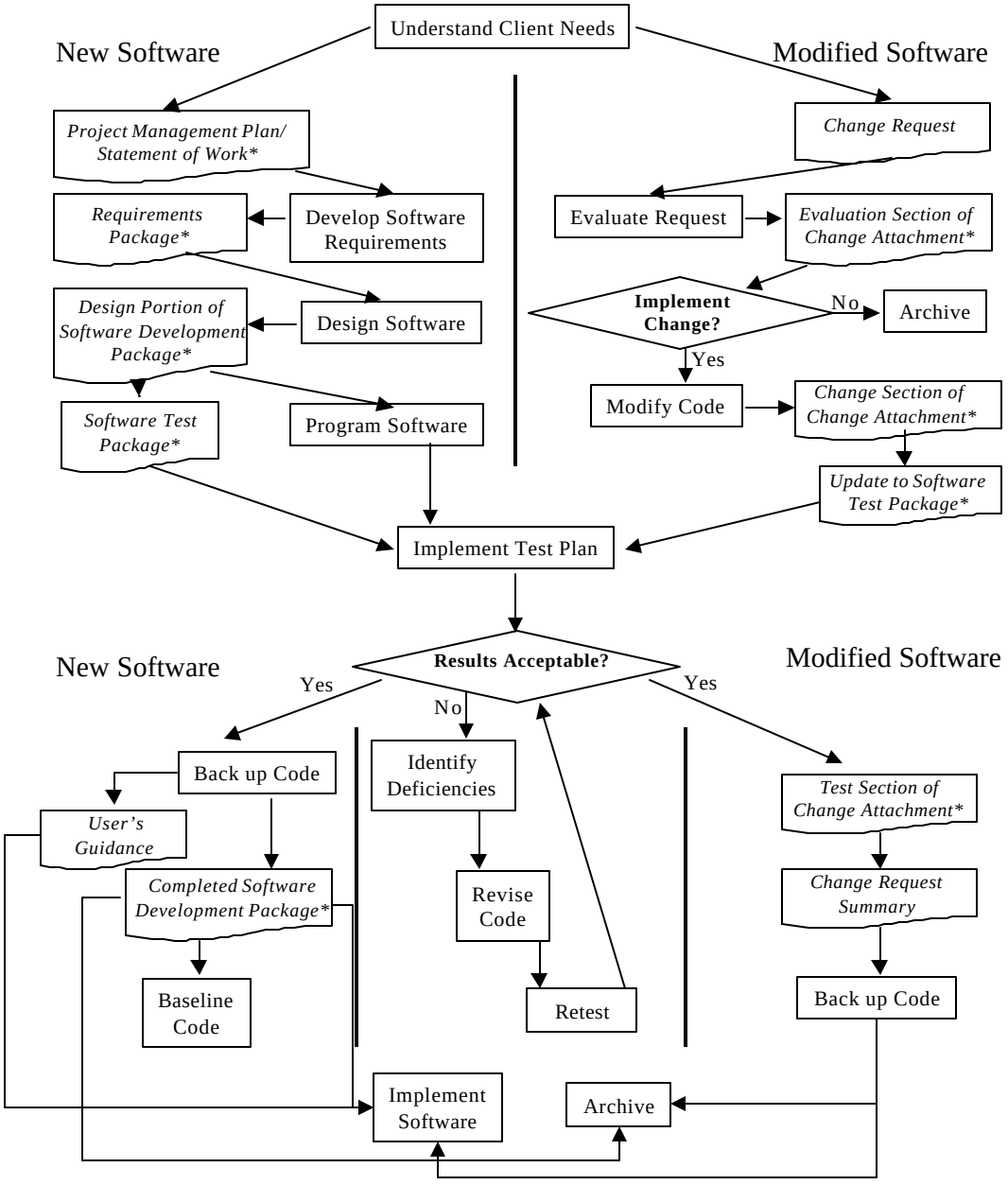

Figure 5.1 Ensuring Quality in the Environmental Software Development Process (* indicates quality review stage; box with wavy bottom line and italics font indicates document versus activity)

Table 5.1 Relationship of PNNL Environmental Software Development Process to Quality Assurance Requirements (OCRWM 1995; EPA 1997)

OCRWM Quality Assurance Requirement[a]	**EPA Essential Element of Information[b]**	**Environmental Software Process Equivalent (Section)**
	4—System Implementation Plan	Project Management Plan or Statement of Work
I.2.5A Functional Requirements Information Documentation; I.2.5C Requirements and Design Documentation	5—System Detailed Requirements Document	Requirements Package
I.2.1 Software Life Cycles, Baselines (see Appendix C), and Controls	6—Software Management Plan	Project Management Plan or Statement of Work and Gelston et al. (1998)
I.2.2 Software Verification[c] and Software Validation; I.2.4 Software Validation[d]	7—Software Test and Acceptance Plan	Software Test Package
I.2.3 Software Verification; I.2.5C Requirements and Design Information Documentation	8—Software Design Document	Design Portion of Software Development Package
I.2.6A Configuration Identification		Completed Software Development Package
I.2.6B Configuration Control; I.2.6C Configuration Status; I.2.7 Defect Reporting and Resolution[e]	9—Software Maintenance Document	Modification Documentation
	10—Software Operations Document	User's Guidance and Training
I.2.5B User Information Documentation	11—Software User's Reference Guide	User's Guidance and Training
	12—System Integration Test Reports	Software Test Package

(a) Note that OCRWM requirement I.2.8, Control of the Use of Software, is the responsibility of the OCRWM-related client.

(b) Elements 1 through 3 are generally completed by clients in EPA before contract initiation with the project team.

(c) Verification includes informal code testing by software engineers (see Appendix C) to ensure that code functions as required.

(d) Validation includes testing by those other than the software engineers who developed the code to provide an independent confirmation that software functions as required.

(e) Note that some changes requested by clients may not be made in the software unless funding has been allocated for such modifications.

A. General Requirements Analysis
--Documented in
_____Statement of Work (stored in project file; see Gene Whelan, Gariann Gelston, or current Integration Leader)
--Contains information on (all of the following)
_____problem description
_____deliverables
_____project team
_____capabilities to be used
_____restrictions
_____difficulties envisioned
_____compatibilities with existing software/hardware
_____scope of the project

B. Specific Requirements Analysis
--Documented in
_____requirements section of documentation (PNNL-11914, Volume 13, Section 2.0)
--Contains information on (all of the following)
_____purpose of the software
_____structure of the software
_____hardware and software requirements
_____input and output requirements
_____scientific basis
_____assumptions
_____limitations
_____post-October 31 requirements

C. Design Documentation
--Documented in
_____design portion of documentation (PNNL-11914, Volume 13, Section 3.0)
_____team task plans/Project Management Plan (stored in project file; see Gene Whelan, Gariann Gelston, or current Integration Leader)
--Contains information on (all of the following)
_____code type and description
_____development team members
_____specifications
_____logic diagrams
_____"help" descriptions
_____methods to ensure consistency in components
_____mathematical formulations
_____need for pre/post-processors
_____post-October 31 design elements

D. Development Documentation
--Documented in
_____Specifications Document (PNNL-11914, Volume 8)
_____Quality Assurance Archive (see Gariann Gelston or current Integration Leader)
--Contains information on (all of the following)
_____baseline hard copy of the source code
_____diskette copy
_____name of computer language(s) used

E. Testing Documentation
--Documented in
_____test plan that meets quality assurance requirements (PNNL-11914, Volume 13, Section 4.0)
--Contains information on (all of the following)
_____description of software
_____testing scope
_____relationship between test cases and requirements
_____test activity description
_____hardware and software needed to implement plan
_____test case specifications
_____expected results

Figure 5.2 Quality Assurance Implementation Checklist for the Chemical Properties Processor

F. User's Guidance
--Documented in
_____hard copy printout of user's guidance for system (PNNL-11914, Volume 11)
--Contains information on (all of the following)
_____description of software
_____description of use of user interface
_____mathematical formulations
_____example problems
_____explanation of modules included

G. General Quality Assurance Documentation
--Documented in
_____Quality Assurance Program Document (PNNL-11880)
_____Quality Assurance Software-Specific Checklist (PNNL-11914, Volume 13, Section 5.0)
--Contains information on (all of the following)
_____purpose of quality assurance program
_____client-specified activities
_____activities required to ensure quality in software

H. Quality Assurance Archive
--Documented in
_____hard copy files (see Gariann Gelston or current Integration Leader)
_____back up disk files in multiple storage locations (see Gariann Gelston or current Integration Leader)
--Contains information on (all of the following)
_____all quality assurance documentation
_____client correspondence regarding software
_____modifications made to baselined software
_____disk copy back ups
_____reproducibility of code (check code for comments)

Completed by _____ Date _____

Approved by
System/Module Manager _____ Date _____

Figure 5.2 Quality Implementation Checklist (contd)

6.0 References

Documentation for the FRAMES-HWIR Technology Software System

Volume 1: Overview of the FRAMES-HWIR Technology Software System. 1998. Pacific Northwest National Laboratory, Richland, Washington.

Volume 2: System User Interface Documentation. 1998. Pacific Northwest National Laboratory, Richland, Washington.

Volume 3: Distribution Statistics Processor Documentation. 1998. TetraTech, Lafayette, California.

Volume 4: Site Definition Processor Documentation. 1998. Pacific Northwest National Laboratory, Richland, Washington.

Volume 5: Computational Optimization Processor Documentation. 1998. TetraTech, Lafayette, California.

Volume 6: Multimedia Multipathway Simulation Processor Documentation. 1998. Pacific Northwest National Laboratory, Richland, Washington.

Volume 7: Exit Level Processor Documentation. 1998. Pacific Northwest National Laboratory, Richland, Washington.

Volume 8: Specifications. 1998. Pacific Northwest National Laboratory, Richland, Washington.

Volume 9: Software Development and Testing Strategies. 1998. Pacific Northwest National Laboratory, Richland, Washington.

Volume 10: Facilitating Dynamic Link Libraries. 1998. Pacific Northwest National Laboratory, Richland, Washington.

Volume 11: User's Guidance. 1998. Pacific Northwest National Laboratory, Richland, Washington.

Volume 12: Dictionary. 1998. Pacific Northwest National Laboratory, Richland, Washington
.
Volume 13: Chemical Properties Processor Documentation. 1998. Pacific Northwest National Laboratory, Richland, Washington.

Volume 14: Site Layout Processor Documentation. 1998. Pacific Northwest National Laboratory, Richland, Washington.

Volume 15: Risk Visualization Processor Documentation. 1998. Pacific Northwest National Laboratory, Richland, Washington.

Quality Assurance Program Document

Gelston, G. M., R. E. Lundgren, J. P. McDonald, and B. L. Hoopes. 1998. *An Approach to Ensuring Quality in Environmental Software*. PNNL-11880, Pacific Northwest National Laboratory, Richland, Washington.

Additional References

Marin, C., and Z. Saleem. 1997. *A Preliminary Framework for Finite-Source Multimedia, Multipathway and Multireceptor Risk Assessment (3MRA)*. Draft, October 1997, U.S. Environmental Protection Agency, Office of Solid Waste, Washington, D.C.

Office of Civilian Radioactive Waste Management (OCRWM). 1995. *Quality Assurance Requirements and Description, Software*. U.S. Department of Energy, Washington, D.C.

U.S. Environmental Protection Agency (EPA). 1997. *System Design and Development Guidance*. EPA Directive Number 2182, Washington, D.C.

Whelan, G., K. J. Castleton, J. W. Buck, G. M. Gelston, B. L. Hoopes, M. A. Pelton, D. L. Strenge, and R. N. Kickert. 1997. *Concepts of a Framework for Risk Analysis in Multimedia Environmental Systems (FRAMES)*. PNNL-11748, Pacific Northwest National Laboratory, Richland, Washington.

Appendix A

Additional Testing Information

Appendix A
Additional Testing Information

A.1 Setting Up Test Cases

To set up a test case for the Chemical Properties Processor (CPP), follow these steps:

1) Open the Windows® Explorer and get to the test bed directory, choosing which subdirectory to use based on compiler (in essence, mscpp for the Microsoft® C++ compiler and bccpp for the Borland® C++ compiler).
2) Make a copy of the file run.bat.
3) Right-mouse-click on the copy and choose Rename, naming it for the test case you will be running (for example, CPP_01C.bat, CPP_02C.bat, etc.). .
4) Right-mouse-click on the new file, and choose Edit.
5) Change the last word on the second line of the file commands to state the test file output (CPP_01C.out, CPP_01C.out, etc.).
6) Change the last word of the third line of the file commands to state the test file (CPP_01test, CPP_02test, etc.). Do not use the extension .tst in this command. Save and exit the file.
7) Make a copy of the file example.tst.
8) Right-mouse-click on the copy and choose Rename, naming it for the test case you will be running (for example, CPP_01Ctest.tst, CPP_02test.tst, etc.). In this case, DO use the extention .tst. Make sure the name matches the word used on the command line in step 5.
9) Open the new file using the Notepad. Edit the file to match your test case as follows:
   - A call for a function or subroutine is simply the name of the function or subroutine, followed on the line directly below by any input data needed (see the Specifications for additional details). Input values are separated by a comma.
   - Subroutine ChemPath comes first, followed by a line on which is listed the directory path for the input data tables.
   - ChemEnv comes next, followed by a line on which are listed the temperature, pH value, type of medium, and Foc value.
   - Subroutine ChemCASID follows; its value is on the line directly below it.
   - Data for any of the ChemPath, ChemEnv, or ChemCASID parameters can be modified, as long as they match the values available in the data tables.
   - For Subroutine ChemInfo, the count for number of chemicals available in the data tables starts at 0 rather than 1. For example, if the chemicals listed in either the Organic Chemical Properties data table or the Metals/Inorganic Chemical Properties data table totaled 6, the returned value from calling ChemInfo would be 5.
10) For each subroutine or function called in the file, after the name, add a comma and a semicolon then a comment as to the result expected. This makes it easier to check results later.

A.2 Example Test Input File for C++ Compilers

The file below is an example of how the HWIRCP.DLL functions are tested without having to recompile a new program for every test. The test program reads a text file that contains the function names and parameter (if any), then makes the appropriate call to the HWIRCP.DLL. In the file below, each function call is made up of at most two lines and at least one. If a function has no parameters passed to it, then the second line is not needed. For example, the NumChem function in the HWIRCP.DLL requires no parameters, so its entry in the test script would be a single line with NumChem in it. The ChemPath function takes one parameter that is the path the HWIRCP.DLL is to use in locating the chemical database. So a test of ChemPath contains two lines—the first is the function name ChemPath, and the second line is the path ..\CPPTest as below.

```
ChemPath
..\CPPTest
ChemEnv
293.15,7.0,Soil,1.0
ChemCASID
1-123-123
pause
ChemSSF
cpp15c.ssf
NumChem, ; expected result 7
ChemInfo, ; expected result 0 Acenaphthene 83-32-9
0
ChemInfo, ; expected result 1 Acetamide 60-35-5
1
ChemInfo, ; expected result 2 Acetic Acid 64-19-7
2
ChemInfo, ; expected result 3 Phosphine 83-32-10
3
ChemInfo, ; expected result 4 Mercury 1-123-123
4
ChemInfo, ; expected result 5 Lead 2-123-123
5
ChemInfo, ; expected result 6 Uranium 3-123-123
6
ChemCASID, ; Checking Organic Calculations
60-35-5
ChemSMILES
ChemMolWt, ; expected result 2
ChemADiff, ; expected result 36.23172238056
ChemVol, ; expected result 59.95804
ChemDen, ; expected result 0.033357
ChemWDiff, ; expected result 0.0287637
ChemVP, ; expected result -2.36686
ChemSol, ; expected result
```

ChemHLC, ; expected result
ChemKow, ; expected result
ChemKoc, ; expected result
ChemKd, ; expected result
ChemCASID
83-32-9
ChemSMILES
ChemMolWt, ; expected result 1
ChemADiff, ; expected result 20.84722216
ChemVol, ; expected result 141.6415
ChemDen, ; expected result 0.00706001
ChemWDiff, ; expected result 1.21758975
ChemVP, ; expected result -2.8725465
ChemSol, ; expected result
ChemHLC, ; expected result
ChemKow, ; expected result
ChemKoc, ; expected result
ChemKd, ; expected result
ChemCASID
64-19-7
ChemSMILES
ChemMolWt, ; expected result 3
ChemADiff, ; expected result 41.718629
ChemVol, ; expected result 57.15804
ChemDen, ; expected result 0.05248605
ChemWDiff, ; expected result 0.0301727
ChemVP, ; expected result 0.0298823
ChemSol, ; expected result
ChemHLC, ; expected result
ChemKow, ; expected result
ChemKoc, ; expected result
ChemKd, ; expected result
ChemCASID
83-32-10
ChemSMILES
ChemMolWt, ; expected result 9
ChemADiff, ; expected result 20.8472216
ChemVol, ; expected result 141.6415
ChemDen, ; expected result0.0633407
ChemWDiff, ; expected result 1.21758975
ChemVP, ; expected result -2.8725465
ChemSol, ; expected result
ChemHLC, ; expected result
ChemKow, ; expected result
ChemKoc, ; expected result
ChemKd, ; expected result
ChemCASID , ; Checking Inorganic Calculations

1-123-123
ChemSol, ; expected result
ChemMolWt, ; expected result 56
ChemKd, ; expected result
ChemCASID
2-123-123
ChemSol, ; expected result
ChemMolWt, ; expected result 75
ChemKd, ; expected result
ChemCASID
3-123-123
ChemSol, ; expected result
ChemMolWt, ; expected result 238
ChemKd, ; expected result
ChemCASID, ; Checking Different Distributions through the AerBio.csv
1-123-123,
ChemAerBio
ChemAerBio
ChemAerBio
ChemAerBio
ChemCASID
2-123-123,
ChemAerBio
ChemAerBio
ChemAerBio
ChemAerBio
ChemCASID
3-123-123,
ChemAerBio
ChemAerBio
ChemAerBio
ChemAerBio
ChemCASID
83-32-9,
ChemAerBio
ChemAerBio
ChemAerBio
ChemAerBio
ChemCASID
60-35-5,
ChemAerBio
ChemAerBio
ChemAerBio
ChemAerBio
ChemCASID
64-19-7,
ChemAerBio

ChemAerBio
ChemAerBio
ChemAerBio
ChemCASID
83-32-10,
ChemAerBio
ChemAerBio
ChemAerBio
ChemAerBio
ChemCASID, ; Checking Transformation products information
60-35-5
ChemHyd, ; expected result 29 3
ChemPHyd, ; expected result 0 Acetic Acid 64-19-7 35
0
ChemPHyd, ; expected result 1 Acenaphthene 83-32-9 41
1
ChemPHyd, ; expected result 2 Acetamide 60-35-5 47
2
ChemActBio, ; expected result 11 3
ChemPActBio, ; expected result 0 Acetic Acid 64-19-7 11
0
ChemPActBio, ; expected result 1 Acenaphthene 83-32-9 17
1
ChemPActBio, ; expected result 2 Acetamide 60-35-5 23
2
ChemAnaRed, ; expected result 11 3
ChemPAnaRed, ; expected result 0 Acetic Acid 64-19-7 11
0
ChemPAnaRed, ; expected result 1 Acenaphthene 83-32-9 17
1
ChemPAnaRed, ; expected result 2 Acetamide 60-35-5 23
2
ChemAnaBio, ; expected result 156 3
ChemPAnaBio, ; expected result 0 Acetic Acid 64-19-7 180
0
ChemPAnaBio, ; expected result 1 Acenaphthene 83-32-9 186
1
ChemPAnaBio, ; expected result 2 Acetamide 60-35-5 192
2
ChemSO4Bio, ; expected result 156 3
ChemPSO4Bio, ; expected result 0 Acetic Acid 64-19-7 180
0
ChemPSO4Bio, ; expected result 1 Acenaphthene 83-32-9 186
1
ChemPSO4Bio, ; expected result 2 Acetamide 60-35-5 192
2
ChemMetBio, ; expected result 156 3

ChemPMetBio, ; expected result 0 Acetic Acid 64-19-7 180
0
ChemPMetBio, ; expected result 1 Acenaphthene 83-32-9 186
1
ChemPMetBio, ; expected result 2 Acetamide 60-35-5 192
2
ChemCASID, ; Checking Transformation products information
83-32-9
ChemHyd, ; expected result 28 3
ChemPHyd, ; expected result 0 Acetamide 60-35-5 34
0
ChemPHyd, ; expected result 1 Acetic Acid 64-19-7 40
1
ChemPHyd, ; expected result 2 Acenaphthene 83-32-9 46
2
ChemActBio, ; expected result 10 3
ChemPActBio, ; expected result 0 Acetamide 60-35-5 10
0
ChemPActBio, ; expected result 1 Acetic Acid 64-19-7 16
1
ChemPActBio, ; expected result 2 Acenaphthene 83-32-9 22
2
ChemAnaRed, ; expected result 10 3
ChemPAnaRed, ; expected result 0 Acetamide 60-35-5 10
0
ChemPAnaRed, ; expected result 1 Acetic Acid 64-19-7 16
1
ChemPAnaRed, ; expected result 2 Acenaphthene 83-32-9 22
2
ChemAnaBio, ; expected result 155 3
ChemPAnaBio, ; expected result 0 Acetamide 60-35-5 179
0
ChemPAnaBio, ; expected result 1 Acetic Acid 64-19-7 185
1
ChemPAnaBio, ; expected result 2 Acenaphthene 83-32-9 191
2
ChemSO4Bio, ; expected result 155 3
ChemPSO4Bio, ; expected result 0 Acetamide 60-35-5 179
0
ChemPSO4Bio, ; expected result 1 Acetic Acid 64-19-7 185
1
ChemPSO4Bio, ; expected result 2 Acenaphthene 83-32-9 191
2
ChemMetBio, ; expected result 155 3
ChemPMetBio, ; expected result 0 Acetamide 60-35-5 179
0
ChemPMetBio, ; expected result 1 Acetic Acid 64-19-7 185

1
ChemPMetBio, ; expected result 2 Acenaphthene 83-32-9 191
2
ChemCASID, ; Checking Transformation products information
64-19-7
ChemHyd, ; expected result 30 3
ChemPHyd, ; expected result 0 Acenaphthene 83-32-9 42
0
ChemPHyd, ; expected result 1 Acetamide 60-35-5 42
1
ChemPHyd, ; expected result 2 Acetic Acid 64-19-7 48
2
ChemActBio, ; expected result 12 3
ChemPActBio, ; expected result 0 Acenaphthene 83-32-9 12
0
ChemPActBio, ; expected result 1 Acetamide 60-35-5 18
1
ChemPActBio, ; expected result 2 Acetic Acid 64-19-7 24
2
ChemAnaRed, ; expected result 12 3
ChemPAnaRed, ; expected result 0 Acenaphthene 83-32-9 12
0
ChemPAnaRed, ; expected result 1 Acetamide 60-35-5 18
1
ChemPAnaRed, ; expected result 2 Acetic Acid 64-19-7 24
2
ChemAnaBio, ; expected result 157 3
ChemPAnaBio, ; expected result 0 Acenaphthene 83-32-9 181
0
ChemPAnaBio, ; expected result 1 Acetamide 60-35-5 187
1
ChemPAnaBio, ; expected result 2 Acetic Acid 64-19-7 193
2
ChemSO4Bio, ; expected result 157 3
ChemPSO4Bio, ; expected result 0 Acenaphthene 83-32-9 181
0
ChemPSO4Bio, ; expected result 1 Acetamide 60-35-5 187
1
ChemPSO4Bio, ; expected result 2 Acetic Acid 64-19-7 193
2
ChemMetBio, ; expected result 157 3
ChemPMetBio, ; expected result 0 Acenaphthene 83-32-9 181
0
ChemPMetBio, ; expected result 1 Acetamide 60-35-5 187
1
ChemPMetBio, ; expected result 2 Acetic Acid 64-19-7 193
2

ChemCASID , ;Checking Transformation products information
1-123-123
ChemHyd, ; expected result 25 3
ChemPHyd, ; expected result 0 Lead 2-123-123 31
0
ChemPHyd, ; expected result 1 Uranium 3-123-123 13
1
ChemPHyd, ; expected result 2 Mercury 1-123-123 43
2
ChemActBio, ; expected result 7 3
ChemPActBio, ; expected result 0 Lead 2-123-123 7
0
ChemPActBio, ; expected result 1 Uranium 3-123-123 13
1
ChemPActBio, ; expected result 2 Mercury 1-123-123 19
2
ChemAnaRed, ; expected result 7 3
ChemPAnaRed, ; expected result 0 Lead 2-123-123 7
0
ChemPAnaRed, ; expected result 1 Uranium 3-123-123 13
1
ChemPAnaRed, ; expected result 2 Mercury 1-123-123 19
2
ChemAnaBio, ; expected result 152 3
ChemPAnaBio, ; expected result 0 Lead 2-123-123 176
0
ChemPAnaBio, ; expected result 1 Uranium 3-123-123 182
1
ChemPAnaBio, ; expected result 2 Mercury 1-123-123 188
2
ChemSO4Bio, ; expected result 152 3
ChemPSO4Bio, ; expected result 0 Lead 2-123-123 176
0
ChemPSO4Bio, ; expected result 1 Uranium 3-123-123 182
1
ChemPSO4Bio, ; expected result 2 Mercury 1-123-123 188
2
ChemMetBio, ; expected result 152 3
ChemPMetBio, ; expected result 0 Lead 2-123-123 176
0
ChemPMetBio, ; expected result 1 Uranium 3-123-123 182
1
ChemPMetBio, ; expected result 2 Mercury 1-123-123 188
2
ChemCASID, ; Checking Transformation products information
2-123-123
ChemHyd, ; expected result 26 3

ChemPHyd, ; expected result 0 Uranium 3-123-123 32
0
ChemPHyd, ; expected result 1 Mercury 1-123-123 38
1
ChemPHyd, ; expected result 2 Lead 2-123-123 44
2
ChemActBio, ; expected result 8 3
ChemPActBio, ; expected result 0 Uranium 3-123-123 8
0
ChemPActBio, ; expected result 1 Mercury 1-123-123 14
1
ChemPActBio, ; expected result 2 Lead 2-123-123 20
2
ChemAnaRed, ; expected result 8 3
ChemPAnaRed, ; expected result 0 Uranium 3-123-123 8
0
ChemPAnaRed, ; expected result 1 Mercury 1-123-123 14
1
ChemPAnaRed, ; expected result 2 Lead 2-123-123 20
2
ChemAnaBio, ; expected result 153 3
ChemPAnaBio, ; expected result 0 Uranium 3-123-123 177
0
ChemPAnaBio, ; expected result 1 Mercury 1-123-123 183
1
ChemPAnaBio, ; expected result 2 Lead 2-123-123 189
2
ChemSO4Bio, ; expected result 153 3
ChemPSO4Bio, ; expected result 0 Uranium 3-123-123 177
0
ChemPSO4Bio, ; expected result 1 Mercury 1-123-123 183
1
ChemPSO4Bio, ; expected result 2 Lead 2-123-123 189
2
ChemMetBio, ; expected result 153 3
ChemPMetBio, ; expected result 0 Uranium 3-123-123 177
0
ChemPMetBio, ; expected result 1 Mercury 1-123-123 183
1
ChemPMetBio, ; expected result 2 Lead 2-123-123 189
2
ChemCASID, ; Checking Transformation products information
3-123-123
ChemHyd, ; expected result 27 3
ChemPHyd, ; expected result 0 Mercury 1-123-123 33
0
ChemPHyd, ; expected result 1 Lead 2-123-123 39

1
ChemPHyd, ; expected result 2 Uranium 3-123-123 45
2
ChemActBio, ; expected result 9 3
ChemPActBio, ; expected result 0 Mercury 1-123-123 9
0
ChemPActBio, ; expected result 1 Lead 2-123-123 15
1
ChemPActBio, ; expected result 2 Uranium 3-123-123 21
2
ChemAnaRed, ; expected result 9 3
ChemPAnaRed, ; expected result 0 Mercury 1-123-123 9
0
ChemPAnaRed, ; expected result 1 Lead 2-123-123 15
1
ChemPAnaRed, ; expected result 2 Uranium 3-123-123 21
2
ChemAnaBio, ; expected result 154 3
ChemPAnaBio, ; expected result 0 Mercury 1-123-123 178
0
ChemPAnaBio, ; expected result 1 Lead 2-123-123 184
1
ChemPAnaBio, ; expected result 2 Uranium 3-123-123 190
2
ChemSO4Bio, ; expected result 154 3
ChemPSO4Bio, ; expected result 0 Mercury 1-123-123 178
0
ChemPSO4Bio, ; expected result 1 Lead 2-123-123 184
1
ChemPSO4Bio, ; expected result 2 Uranium 3-123-123 190
2
ChemMetBio, ; expected result 154 3
ChemPMetBio, ; expected result 0 Mercury 1-123-123 178
0
ChemPMetBio, ; expected result 1 Lead 2-123-123 184
1
ChemPMetBio, ; expected result 2 Uranium 3-123-123 190
2
ChemCASID, ;Checking Human and Ecological Benchmarks
60-35-5
ChemHum, ; expected result 5 11 1 17 23 0
ChemMam, ; expected result 5 -2
'mule deer'
ChemMam, ; expected result 11 -2
'short-tailed weasel'
ChemBird, ; expected result 17 -2
'belted kingfisher'

ChemBird, ; expected result 23 -2
'tree swallow'
ChemRep, ; expected result 29 -2
'snapping turtle'
ChemRep, ; expected result 35 -2
'pine snake'
ChemAmph, ; expected result 41 -2
'bullfrog'
ChemAmph, ; expected result 47 -2
'flatwood salamander'
ChemCom, ; expected result 53 -2
'Aquatic Biota(dslvd)'
ChemCom, ; expected result 59 -2
'Plants'
ChemCASID, ;Checking Human and Ecological Benchmarks
83-32-9
ChemHum, ; expected result 4 10 0 16 22 1
ChemMam, ; expected result 4 -1
'mule deer'
ChemMam, ; expected result 10 -1
'short-tailed weasel'
ChemBird, ; expected result 16 -1
'belted kingfisher'
ChemBird, ; expected result 22 -1
'tree swallow'
ChemRep, ; expected result 28 -1
'snapping turtle'
ChemRep, ; expected result 34 -1
'pine snake'
ChemAmph, ; expected result 40 -1
'bullfrog'
ChemAmph, ; expected result 46 -1
'flatwood salamander'
ChemCom, ; expected result 52 -1
'Aquatic Biota(dslvd)'
ChemCom, ; expected result 58 -1
'Plants'
ChemCASID, ;Checking Human and Ecological Benchmarks
64-19-7
ChemHum, ; expected result 6 12 0 18 24 1
ChemMam, ; expected result 6 -3
'mule deer'
ChemMam, ; expected result 12 -3
'short-tailed weasel'
ChemBird, ; expected result 18 -3
'belted kingfisher'
ChemBird, ; expected result 24 -3

'tree swallow'
ChemRep, ; expected result 30 -3
'snapping turtle'
ChemRep, ; expected result 36 -3
'pine snake'
ChemAmph, ; expected result 42 -3
'bullfrog'
ChemAmph, ; expected result 48 -3
'flatwood salamander'
ChemCom, ; expected result 54 -3
'Aquatic Biota(dslvd)'
ChemCom, ; expected result 60 -3
'Plants'
ChemCASID , ;Checking Human and Ecological Benchmarks
1-123-123
ChemHum, ; expected result 1 7 1 13 19 0
ChemMam, ; expected result 1 -1
'mule deer'
ChemMam, ; expected result 7 -1
'short-tailed weasel'
ChemBird, ; expected result 13 -1
'belted kingfisher'
ChemBird, ; expected result 19 -1
'tree swallow'
ChemRep, ; expected result 25 -1
'snapping turtle'
ChemRep, ; expected result 31 -1
'pine snake'
ChemAmph, ; expected result 37 -1
'bullfrog'
ChemAmph, ; expected result 43 -1
'flatwood salamander'
ChemCom, ; expected result 49 -1
'Aquatic Biota(dslvd)'
ChemCom, ; expected result 55 -1
'Plants'
ChemCASID, ;Checking Human and Ecological Benchmarks
2-123-123
ChemHum, ; expected result 2 8 0 14 20 1
ChemMam, ; expected result 2 -2
'mule deer'
ChemMam, ; expected result 8 -2
'short-tailed weasel'
ChemBird, ; expected result 14 -2
'belted kingfisher'
ChemBird, ; expected result 20 -2
'tree swallow'

ChemRep, ; expected result 26 -2
'snapping turtle'
ChemRep, ; expected result 32 -2
'pine snake'
ChemAmph, ; expected result 38 -2
'bullfrog'
ChemAmph, ; expected result 44 -2
'flatwood salamander'
ChemCom, ; expected result 50 -2
'Aquatic Biota(dslvd)'
ChemCom, ; expected result 56 -2
'Plants'
ChemCASID, ;Checking Human and Ecological Benchmarks
3-123-123
ChemHum, ; expected result 3 9 1 15 21 0
ChemMam, ; expected result 3 -3
'mule deer'
ChemMam, ; expected result 9 -3
'short-tailed weasel'
ChemBird, ; expected result 15 -3
'belted kingfisher'
ChemBird, ; expected result 21 -3
'tree swallow'
ChemRep, ; expected result 27 -3
'snapping turtle'
ChemRep, ; expected result 33 -3
'pine snake'
ChemAmph, ; expected result 39 -3
'bullfrog'
ChemAmph, ; expected result 45 -3
'flatwood salamander'
ChemCom, ; expected result 51 -3
'Aquatic Biota(dslvd)'
ChemCom, ; expected result 57 -3
'Plants'
ChemCASID, ;Checking Terrestrial Bioconcentration Factors
60-35-5
ChemSoilTo, ; expected result 5
'exveg'
ChemSoilTo, ; expected result 17
'proveg'
ChemSoilTo, ; expected result 23
'exfruit'
ChemSoilTo, ; expected result 35
'profruit'
ChemSoilTo, ; expected result 41
'root'

ChemSoilTo, ; expected result 53
'grain'
ChemSoilTo, ; expected result 59
'silage'
ChemSoilTo, ; expected result 71
'forage'
ChemSoilTo, ; expected result 107
'earthworm'
ChemSoilTo, ; expected result 113
'invertebrate'
ChemSoilTo, ; expected result 119
'small mammal'
ChemSoilTo, ; expected result 125
'other vertebrate'
ChemAirTo, ; expected result 11
'exveg'
ChemAirTo, ; expected result 29
'exfruit'
ChemAirTo, ; expected result 65
'silage'
ChemAirTo, ; expected result 77
'forage'
ChemRCF, ; expected result 47
ChemBS, ; expected result 101
ChemBTF, ; expected result 83
'milk'
ChemBTF, ; expected result 89
'beef'
ChemBTF, ; expected result 95
'water'
ChemCASID, ;Checking Terrestrial Bioconcentration Factors
83-32-9
ChemSoilTo, ; expected result 4
'exveg'
ChemSoilTo, ; expected result 16
'proveg'
ChemSoilTo, ; expected result 22
'exfruit'
ChemSoilTo, ; expected result 34
'profruit'
ChemSoilTo, ; expected result 40
'root'
ChemSoilTo, ; expected result 52
'grain'
ChemSoilTo, ; expected result 58
'silage'
ChemSoilTo, ; expected result 70

'forage'
ChemSoilTo, ; expected result 106
'earthworm'
ChemSoilTo, ; expected result 112
'invertebrate'
ChemSoilTo, ; expected result 118
'small mammal'
ChemSoilTo, ; expected result 124
'other vertebrate'
ChemAirTo, ; expected result 10
'exveg'
ChemAirTo, ; expected result 28
'exfruit'
ChemAirTo, ; expected result 64
'silage'
ChemAirTo, ; expected result 76
'forage'
ChemRCF, ; expected result 46
ChemBS, ; expected result 100
ChemBTF, ; expected result 82
'milk'
ChemBTF, ; expected result 88
'beef'
ChemBTF, ; expected result 94
'water'
ChemCASID, ;Checking Terrestrial Bioconcentration Factors
64-19-7
ChemSoilTo, ; expected result 6
'exveg'
ChemSoilTo, ; expected result 18
'proveg'
ChemSoilTo, ; expected result 24
'exfruit'
ChemSoilTo, ; expected result 36
'profruit'
ChemSoilTo, ; expected result 42
'root'
ChemSoilTo, ; expected result 54
'grain'
ChemSoilTo, ; expected result 60
'silage'
ChemSoilTo, ; expected result 72
'forage'
ChemSoilTo, ; expected result 108
'earthworm'
ChemSoilTo, ; expected result 114
'invertebrate'

ChemSoilTo, ; expected result 120
'small mammal'
ChemSoilTo, ; expected result 126
'other vertebrate'
ChemAirTo, ; expected result 12
'exveg'
ChemAirTo, ; expected result 30
'exfruit'
ChemAirTo, ; expected result 66
'silage'
ChemAirTo, ; expected result 78
'forage'
ChemRCF, ; expected result 48
ChemBS, ; expected result 102
ChemBTF, ; expected result 84
'milk'
ChemBTF, ; expected result 90
'beef'
ChemBTF, ; expected result 96
'water'
ChemCASID, ;Checking Terrestrial Bioconcentration Factors
1-123-123
ChemSoilTo, ; expected result 1
'exveg'
ChemSoilTo, ; expected result 13
'proveg'
ChemSoilTo, ; expected result 19
'exfruit'
ChemSoilTo, ; expected result 31
'profruit'
ChemSoilTo, ; expected result 37
'root'
ChemSoilTo, ; expected result 49
'grain'
ChemSoilTo, ; expected result 55
'silage'
ChemSoilTo, ; expected result 67
'forage'
ChemSoilTo, ; expected result 103
'earthworm'
ChemSoilTo, ; expected result 109
'invertebrate'
ChemSoilTo, ; expected result 115
'small mammal'
ChemSoilTo, ; expected result 121
'other vertebrate'
ChemAirTo, ; expected result 7

'exveg'
ChemAirTo, ; expected result 25
'exfruit'
ChemAirTo, ; expected result 61
'silage'
ChemAirTo, ; expected result 73
'forage'
ChemRCF, ; expected result 43
ChemBS, ; expected result 97
ChemBTF, ; expected result 79
'milk'
ChemBTF, ; expected result 85
'beef'
ChemBTF, ; expected result 91
'water'
ChemCASID, ;Checking Terrestrial Bioconcentration Factors
2-123-123
ChemSoilTo, ; expected result 2
'exveg'
ChemSoilTo, ; expected result 14
'proveg'
ChemSoilTo, ; expected result 20
'exfruit'
ChemSoilTo, ; expected result 32
'profruit'
ChemSoilTo, ; expected result 38
'root'
ChemSoilTo, ; expected result 50
'grain'
ChemSoilTo, ; expected result 56
'silage'
ChemSoilTo, ; expected result 68
'forage'
ChemSoilTo, ; expected result 104
'earthworm'
ChemSoilTo, ; expected result 110
'invertebrate'
ChemSoilTo, ; expected result 116
'small mammal'
ChemSoilTo, ; expected result 122
'other vertebrate'
ChemAirTo, ; expected result 8
'exveg'
ChemAirTo, ; expected result 25
'exfruit'
ChemAirTo, ; expected result 62
'silage'

ChemAirTo, ; expected result 74
'forage'
ChemRCF, ; expected result 44
ChemBS, ; expected result 98
ChemBTF, ; expected result 80
'milk'
ChemBTF, ; expected result 86
'beef'
ChemBTF, ; expected result 92
'water'
ChemCASID, ;Checking Terrestrial Bioconcentration Factors
3-123-123
ChemSoilTo, ; expected result 3
'exveg'
ChemSoilTo, ; expected result 15
'proveg'
ChemSoilTo, ; expected result 21
'exfruit'
ChemSoilTo, ; expected result 33
'profruit'
ChemSoilTo, ; expected result 39
'root'
ChemSoilTo, ; expected result 51
'grain'
ChemSoilTo, ; expected result 57
'silage'
ChemSoilTo, ; expected result 69
'forage'
ChemSoilTo, ; expected result 105
'earthworm'
ChemSoilTo, ; expected result 111
'invertebrate'
ChemSoilTo, ; expected result 117
'small mammal'
ChemSoilTo, ; expected result 123
'other vertebrate'
ChemAirTo, ; expected result 9
'exveg'
ChemAirTo, ; expected result 27
'exfruit'
ChemAirTo, ; expected result 63
'silage'
ChemAirTo, ; expected result 75
'forage'
ChemRCF, ; expected result 45
ChemBS, ; expected result 99
ChemBTF, ; expected result 81

'milk'
ChemBTF, ; expected result 87
'beef'
ChemBTF, ; expected result 93
'water'
ChemCASID, ; checking aquatic bioconcentration factors
60-35-5
ChemMT, ; expected result 5
ChemABF, ; expected result 11
'plant'
ChemABF, ; expected result 17
'finfish'
ChemABF, ; expected result 23
'shellfish'
ChemABF, ; expected result 29
'phytoplankton'
ChemABF, ; expected result 35
'zooplankton'
ChemABF, ; expected result 41
'benthos category 1'
ChemABF, ; expected result 47
'benthos category 2'
ChemABF, ; expected result 53
'invertebrates'
ChemABF, ; expected result 59
'T3 finfish'
ChemABF, ; expected result 65
'T4 finfish'
ChemCASID, ; checking aquatic bioconcentration factors
83-32-9
ChemMT, ; expected result 4
ChemABF, ; expected result 10
'plant'
ChemABF, ; expected result 16
'finfish'
ChemABF, ; expected result 22
'shellfish'
ChemABF, ; expected result 28
'phytoplankton'
ChemABF, ; expected result 34
'zooplankton'
ChemABF, ; expected result 40
'benthos category 1'
ChemABF, ; expected result 46
'benthos category 2'
ChemABF, ; expected result 52
'invertebrates'

ChemABF, ; expected result 58
'T3 finfish'
ChemABF, ; expected result 64
'T4 finfish'
ChemCASID, ; checking aquatic bioconcentration factors
64-19-7
ChemMT, ; expected result 6
ChemABF, ; expected result 12
'plant'
ChemABF, ; expected result 18
'finfish'
ChemABF, ; expected result 24
'shellfish'
ChemABF, ; expected result 30
'phytoplankton'
ChemABF, ; expected result 36
'zooplankton'
ChemABF, ; expected result 42
'benthos category 1'
ChemABF, ; expected result 48
'benthos category 2'
ChemABF, ; expected result 54
'invertebrates'
ChemABF, ; expected result 60
'T3 finfish'
ChemABF, ; expected result 66
'T4 finfish'
ChemCASID, ; checking aquatic bioconcentration factors
1-123-123
ChemMT, ; expected result 1
ChemABF, ; expected result 7
'plant'
ChemABF, ; expected result 13
'finfish'
ChemABF, ; expected result 19
'shellfish'
ChemABF, ; expected result 25
'phytoplankton'
ChemABF, ; expected result 31
'zooplankton'
ChemABF, ; expected result 37
'benthos category 1'
ChemABF, ; expected result 43
'benthos category 2'
ChemABF, ; expected result 49
'invertebrates'
ChemABF, ; expected result 55

'T3 finfish'
ChemABF, ; expected result 61
'T4 finfish'
ChemCASID, ; checking aquatic bioconcentration factors
2-123-123
ChemMT, ; expected result 2
ChemABF, ; expected result 8
'plant'
ChemABF, ; expected result 14
'finfish'
ChemABF, ; expected result 20
'shellfish'
ChemABF, ; expected result 26
'phytoplankton'
ChemABF, ; expected result 32
'zooplankton'
ChemABF, ; expected result 38
'benthos category 1'
ChemABF, ; expected result 44
'benthos category 2'
ChemABF, ; expected result 50
'invertebrates'
ChemABF, ; expected result 56
'T3 finfish'
ChemABF, ; expected result 62
'T4 finfish'
ChemCASID, ; checking aquatic bioconcentration factors
3-123-123
ChemMT, ; expected result 3
ChemABF, ; expected result 9
'plant'
ChemABF, ; expected result 15
'finfish'
ChemABF, ; expected result 21
'shellfish'
ChemABF, ; expected result 27
'phytoplankton'
ChemABF, ; expected result 33
'zooplankton'
ChemABF, ; expected result 39
'benthos category 1'
ChemABF, ; expected result 45
'benthos category 2'
ChemABF, ; expected result 51
'invertebrates'
ChemABF, ; expected result 57
'T3 finfish'

ChemABF, ; expected result 63
'T4 finfish'
Pause
Stop